*FORM PTO-1390 OFFICE (REV 11-2000)	U.S DEPARTMENT OF COMMERCE PATENT AND TRADEMARK	ATTORNEY'S DOCKET NUMBER 514592000100

TRANSMITTAL LETTER TO THE UNITED STATES DESIGNATED/ELECTED OFFICE (DO/EO/US) CONCERNING A FILING UNDER 35 U.S.C. § 371

U.S. APPLICATION NO. (If known, see 37 CFR 1.5)

10/ 088949

INTERNATIONAL APPLICATION NO.	INTERNATIONAL FILING DATE	PRIORITY DATE CLAIMED
PCT/US00/26631	September 28, 2000	September 29, 1999

TITLE OF INVENTION
SYSTEM TO COORDINATE THE EXECUTION OF A PLURALITY OF SEPARATE COMPUTER SYSTEMS TO EFFECTUATE A PROCESS (AMENDED IN SEARCH REPORT)

APPLICANT(S) FOR DO/EO/US
Anna PETROVSKAYA

Applicant herewith submits to the United States Designated/Elected Office (DO/EO/US) the following items and other information:

1. ☒ This is a **FIRST** submission of items concerning a filing under 35 U.S.C. 371.

2. ☐ This is a **SECOND** or **SUBSEQUENT** submission of items concerning a filing under 35 U.S.C. 371.

3. ☐ This is an express request to begin national examination procedures (35 U.S.C. 371(f)). The submission must include items (5), (6), (9) and (21) indicated below.

4. ☐ The US has been elected by the expiration of 19 months from the priority date (PCT Article 31).

5. ☒ A copy of the International Application as filed (35 U.S.C. 371(c)(2))
   a. ☐ is attached hereto (required only if not communicated by the International Bureau).
   b. ☐ has been communicated by the International Bureau.
   c. ☒ is not required, as the application was filed in the United States Receiving Office (RO/US).

6. ☐ An English language translation of the International Application as filed (35 U.S.C. 371(c)(2)).
   a. ☐ is attached hereto.
   b. ☐ has been previously submitted under 35 U.S.C. 154(d)(4).

7. ☒ Amendments to the claims of the International Application under PCT Article 19 (35 U.S.C. 371(c)(3)).
   a. ☐ are attached hereto (required only if not communicated by the International Bureau).
   b. ☐ have been communicated by the International Bureau.
   c. ☐ have not been made; however, the time limit for making such amendments has NOT expired.
   d. ☒ have not been made and will not be made.

8. ☐ An English language translation of the amendments to the claims under PCT Article 19 (35 U.S.C. 371(c)(3)).

9. ☒ An oath or declaration of the inventor(s) (35 U.S.C. 371(c)(4)) - 2 pages.

10. ☐ An English language translation of the annexes to the International Preliminary Examination Report under PCT Article 36 (35 U.S.C. 371(c)(5)).

Items 11. to 16. below concern document(s) or information included:

11. ☐ An Information Disclosure Statement under 37 CFR 1.97 and 1.98.

12. ☐ An assignment document for recording. A separate cover sheet in compliance with 37 CFR 3.28 and 3.31 is included.

13. ☐ A FIRST preliminary amendment.

14. ☐ A SECOND or SUBSEQUENT preliminary amendment.

15. ☐ A substitute specification.

16. ☐ A change of power of attorney and/or address letter.

17. ☐ A computer-readable form of the sequence listing in accordance with PCT Rule 13ter.2 and 35 U.S.C. 1.821 - 1.825.

18. ☐ A second copy of the published international application under 35 U.S.C. 154(d)(4).

19. ☐ A second copy of the English language translation of the international application under 35 U.S.C. 154(d)(4).

20. ☒ Other items or information: Application Data Sheet (2 pages) and Return receipt postcard.

CERTIFICATE OF MAILING BY "EXPRESS MAIL"

Express Mail Label No.: EV093210707US   Date of Deposit: March 22, 2002

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. § 1.10 on the date indicated above and is addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231.

Tamara Alcaraz

U.S. APPLICATION NO. (° 10/ 088949	INTERNATIONAL APPLICATION NO. PCT/US00/26631	ATTORNEY'S DOCKET NUMBER: 514592000100

21. ☒ The following fees are submitted:

BASIC NATIONAL FEE (37 CFR 1.492(a)(1)-(5)):

CALCULATIONS PTO USE ONLY

Neither international preliminary examination fee (37 CFR 1.482) nor international search fee (37 CFR 1.445(a)(2)) paid to USPTO and International Search Report not prepared by the EPO or JPO.........................$1,040.00

International preliminary examination fee (37 CFR 1.482) not paid to USPTO but International Search Report prepared by the EPO or JPO......................$890.00

International preliminary examination fee (37 CFR 1.482) not paid to USPTO but international search fee (37 CFR 1.445(a)(2)) paid to USPTO.........................$740.00

International preliminary examination fee (37 CFR 1.482) paid to USPTO but all claims did not satisfy provision of PCT Article 33(1)-(4) $710.00

International preliminary examination fee (37 CFR 1.482) paid to USPTO and all claims satisfied provisions of PCT Article 33(1)-(4) $100.00

			ENTER APPROPRIATE BASIC FEE AMOUNT =	$890.00	
Surcharge of **$130.00** for furnishing the oath or declaration later than ☐ 20 ☒ 30 months from the earliest claimed priority date (37 CFR 1.492(e)).				$0.00	
CLAIMS	NUMBER FILED	NUMBER EXTRA	RATE		
Total claims	35 - 20 =	15	x $18.00	$270.00	
Independent claims	2 - 3 =	0	x $84.00	$0.00	
MULTIPLE DEPENDENT CLAIM(S) (if applicable)			+ $280.00	$0.00	
			TOTAL OF ABOVE CALCULATIONS =	$1160.00	
☒☐Applicant claims small entity status. See 37 CFR 1.27. The fees indicated above are reduced by ½.				$580.00	
			SUBTOTAL =	$580.00	
Processing fee of **$130.00** for furnishing the English translation later than ☐ 20 ☐ 30 months from the earliest claimed priority date (37 CFR 1.492(f)).			+	$0.00	
			TOTAL NATIONAL FEE =	$0.00	
Fee for recording the enclosed assignment (37 CFR 1.21(h)). The assignment must be accompanied by an appropriate cover sheet (37 CFR 3.28, 3.31). **$40.00 per property**			+	$0.00	
			TOTAL FEES ENCLOSED =	$0.00	
				Amount to be refunded:	$0.00
				charged:	$580.00

a. ☐ A check in the amount of $ to cover the above fees is enclosed.

b. ☒ Please charge my **Deposit Account No. 03-1952** in the amount of $580.00 to cover the above fees. A duplicate copy of this sheet is enclosed.

c. ☒ The Commissioner is hereby authorized to charge any additional fees that may be required, or credit any overpayment to **Deposit Account No. 03-1952**. A duplicate copy of this sheet is enclosed.

d. ☐ Fees are to be charged to a credit card. **WARNING:** Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

NOTE: Where an appropriate time limit under 37 CFR 1.494 or 1.495 has not been met, a petition to revive (37 CFR 1.137(a) or (b)) must be filed and granted to restore the application to pending status.

SEND ALL CORRESPONDENCE TO:

Alan S. Hodes
Morrison & Foerster LLP
755 Page Mill Road
Palo Alto, California 94304-1018

SIGNATURE

Alan S. Hodes
Registration No. 38,185

10/088949

SYSTEM FOR DEVELOPMENT AND MAINTENANCE OF SOFTWARE SOLUTIONS FOR EXECUTION ON DISTRIBUTED COMPUTER SYSTEMS

RELATED APPLICATIONS

5   This application claims priority under 35 U.S.C. §119(e) to provisional patent application serial no. 60/156,809, filed September 29, 1999.

COPYRIGHT AUTHORIZATION

TECHNICAL FIELD

        The present invention is in the field of development and maintenance of

15   software solutions. More particularly, the invention relates to a tool that somewhat systematizes and automates the process of developing and maintaining such software solutions for execution on distributed computer systems.

BACKGROUND

        We live during exiting times of hi-tech revolution that deeply affects many

20   aspects of our society. In particular it has changed how companies compete with one another. No matter what industry sector, today's competitive edge comes from mastering emerging technologies ahead of the rivals. For example, a restaurant's web page with a current menu gives a competitive advantage over other restaurants. In the software business, customers prefer companies that provide documentation, customer

25   support and sales on-line. While the revolution has been triggered by new communication technologies (popularly known as "the Internet"), the rapid pace has been sustained by using software rather than hardware to implement many features of emerging technologies quickly and inexpensively. This rapid pace of technology creates an ever-changing environment. The environment is so dynamic that many

30   times features designed into software at one point of time are no longer sufficient by

the time the software gets to the market. Consequently, new technologies
systematically come out without being fully accustomed to the environment in which
they are to be used. However, competition is pushing many companies to make the
best use of emerging technologies by integrating and extending them to the point
5   where they provide full solutions. We call these solutions "in-house solutions" as
they are developed inside of a company and customized for its own use. We shall
distinguish in-house solutions from commercial solutions that are made for sale.

In-house solutions may be built to implement business methods or to improve
operations. Business method in-house solutions implement entire business methods
10   such as selling books online, recording CDs to customer order or providing Internet
services to clients. While not every business method can be implemented by an in-
house solution, many businesses employ in-house solutions to improve internal or
external operations. Internally focused in-house solutions implement employee or
equipment related processes. For example, an internally focused in-house solution
15   may implement an "incoming process" used by companies whenever they hire a new
employee. Externally focused in-house solutions implement processes targeted at
clients, suppliers or partners. While business method in-house solutions enable new
types of businesses to function, operations oriented in-house solutions cut costs and
delays by streamlining operations. For over a decade, in-house solutions have been
20   enabling companies to survive and compete in today's hi-tech revolution conditions.

Conventional tools for implementing "in-house solutions" are now discussed.
On the low technical level, in-house development and maintenance are very similar to
their commercial counterparts. Consequently, in-house developers and administrators
have traditionally used generic tools such as languages, compilers, libraries, version
25   control applications, bug tracking systems and others. However, as it is discussed in
greater detail below, there are some important differences between in-house and
commercial solutions. Because of these differences, enterprises developing and
maintaining in-house solutions have needs that have not been addressed for over a
decade.

30   Let us list and briefly explain some of the problems that arise from the fact
that in-house solutions are not made for a large commercial market. First, the
financial gains from developing such software are significantly lower. While millions

of copies of commercial software could be sold for $100 or more per copy, in-house solutions are considered a success if they save a few million dollars over their lifetime. Thus funding available for in-house solutions is under 1% of funding available for commercial software. Consequently, enterprises can not afford to keep permanent development and support teams working on in-house solutions. Instead, they gather expertise from all over the enterprise for a short period to develop the solution. The support is done on demand by administrators of the solution. While this tactic saves money, it also results in a multitude of problems that we will summarize below.

Since development team does not exist as a stand-alone group, immediate concerns prevail over long-term needs during development cycle. Infrastructure work (such as architecture and documentation) gives way to specific convenience features. Little assessment and plans are made for how solution will evolve with changing needs of enterprise. Since members of the team come from different groups and do not work on the project full-time, they bring with themselves priorities and schedules that come from their other projects. This leads to longer development cycles, because the whole team moves at the pace of its slowest member at any given point in time.

In contrast to a commercial development environment, there is no dedicated technical writer to write proper documentation and to keep track of changes made to software over time. Therefore, in-house solutions are usually poorly or not at all documented. Furthermore, unlike in commercial development, there are no dedicated Quality Assurance engineers. Thus, Quality Assurance is not performed at all or performed by engineers who have little knowledge of the product they are testing. Consequently, very few bugs are discovered prior to the product going live.

In commercial development, the development team is kept intact after the first version of solution goes live. Moreover, the development team trains additional resources to help maintain and support the solution. In contrast, the in-house development team is disbanded after the solution goes live. Improvements and bug fixes of in-house solutions are usually done sporadically by new developers. There is no guarantee that developers extending a solution participated in its initial development. Therefore, no expertise is passed on from the original development team to developers who end up maintaining the solution. This is especially harmful to

3

the quality of the solution because there is no up-to-date documentation. Consequently, expertise and code are not reused during extensions of a solution.

Since many new technologies are integrated in an in-house solution, it is virtually impossible to find an administrator who is experienced in working with all of the technologies involved. Furthermore, administrators who support in-house solutions receive no training and are given no documentation to train themselves. Consequently, not only does it take a long time to debug and fix problems, but also new problems are often introduced via patches made by administrators with little knowledge of the system.

Another drawback of in-house solutions is that they get very little exposure to a technical audience. Since millions of copies of commercial software are typically sold, millions of technical teams comprised of developers and administrators learn about the software as well as install and test it for their own use. This fact is well known to IT departments, who like to let new software float on the market for a few months before installing it on their own systems. Many bugs are usually discovered within the first few months of new software being on the market and developers create jumbo patches that are distributed to customers. In contrast, in-house solutions only enjoy the audience of one development team and a few administrators who maintain the system later. Thus, bugs inherent in the solution may not be discovered until after they have caused extensive damage. While bugs contribute to overall low quality of software, it is in the system security area where bugs may be the most harmful. This is because one small security bug can invalidate the security of the whole system. While commercial solutions enjoy the benefits of collective efforts to discover security bugs and prevent break-ins, security of an in-house solution is left to a small team of developers.

A major purpose of in-house development is to integrate available technologies into a solution that closely fits the needs of the enterprise. As we mentioned earlier, the rapid pace of technology creates an ever-changing environment. Therefore, to stay effective, an in-house solution should evolve dynamically with the changing needs of the enterprise. Unfortunately poor architecture, missing documentation and absent training make extending in-house solutions a slow, error-prone and inefficient process. Since measuring the cost of

inefficiency and low quality is a difficult task, let us turn to an example where they translated into some very visible numbers. On June 10th of 1999 a twenty two-hour outage of eBay's website was followed by a $2 billion drop in the company's market value. The cause of the outage was determined to be an attempt to add new features

5   to an existing in-house system.

To summarize, it is desired that today's enterprise be able to efficiently develop and maintain in-house solutions that are comparable in quality, reliability and security to commercial solutions; that are subject to all the limiting factors imposed on in-house solutions; and that dynamically adapt to changes in enterprise's needs.

10   **SUMMARY**

In accordance with the invention, a system is provided to effectuate steps of a process such as a business process. A core system receives a request by a user to effectuate the process, along with user data upon which it is desired to effectuate the process. A coordinating system causes and coordinates execution of a plurality of

15   target computer system based on the indication of the action and user data, to accomplish effectuation of the process.

                          BRIEF DESCRIPTION OF FIGURES

Fig. 1 is a schematic illustration of the environment in which the invention operates.

20   Fig. 2 illustrates an embodiment in accordance with the invention .

Fig. 3 illustrates a particular detailed implementation of the Fig. 2 embodiment.

Fig. 4 illustrates the Fig. 3 implementation in greater detail.

Fig. 5 illustrates a tree data model usable in accordance with an embodiment

25   of the invention.

Fig. 6 illustrates an example of a specific tree structure.

Figs. 7 and 8 illustrate a method by which an ENGINE processes a Request.

DETAILED DESCRIPTION

We now describe in detail several embodiments in accordance with the invention. The description includes a source code listing included herein as Appendix A. We use *italicized* words to denote elements of our data model and CAPITALIZED words to denote external components, components of our system and subjects. We use typewriter font to denote machine commands, data, file and directory names. In referring to our source code, we will use relative and absolute pathnames. By default, relative pathnames will be relative to /share/Kiki/WF/prod directory.

A schematic representation of our system is depicted in Fig. 1. The box labeled CORE SYSTEM 102 represents the core of the embodiment. Boxes labeled APPLICATION 1 (104a) through APPLICATION N (104n) represent existing applications employed in an enterprise. Boxes labeled DEVELOPER 106, ADMINISTRATOR 108 and USER 110 represent an in-house solution developer, an in-house solution administrator and an in-house solution user respectively. Arrows represent directions of data flows.

To create an in-house solution, DEVELOPER 106 interacts with CORE SYSTEM 102 to define an *Action*. An *Action* is a definition of an in-house solution. It contains information as to which APPLICATIONS 104 are to be involved and what data needs to be collected from USER 110 and passed to APPLICATIONS 104, as well as rules for execution. To use the in-house solution created by DEVELOPER 106, USER 110 interacts with CORE SYSTEM 102 and places a *Request* to run the *Action*. *Request* contains data passed by USER 110 and a reference to the *Action*. When a *Request* has been placed, CORE SYSTEM 102 interacts with APPLICATIONS 1 through N (104a through 104n) specified in *Action* and passes USER 110 data from the *Request* to APPLICATIONS 104 following rules defined in the *Action*. ADMINISTRATOR 108 interacts with CORE SYSTEM 102 to monitor execution of the *Request*, diagnose and troubleshoot problems if they arise. *Action* and *Request* are part of our data model that is discussed in more detail later in this Detailed Description.

Let us look at CORE SYSTEM 102 in more detail (see Fig. 2). We identify the following components of CORE SYSTEM 102: user interface (UI) 202, data store

(STORE) 204 and engine (ENGINE) 206. UI 202 enables exchange of information between DEVELOPER 106, ADMINISTRATOR 108 or USER 110 on one hand and STORE 204 on the other hand. STORE 204 is used to hold *Actions*, *Requests*, execution data, logs, ENGINE 206 state information, and other data necessary for the

5   system to function. ENGINE 206 monitors STORE 204 for new *Action* definitions created by DEVELOPER 106 and new *Requests* posted by USER 110. When ENGINE 206 receives a *Request*, it verifies its consistency with the corresponding *Action* definition. ENGINE 206 then passes USER 110 data from the *Request* to APPLICATIONS 104 according to rules specified in the *Action*. ENGINE 206

10   monitors communications with APPLICATIONS 104 and receives updates on completion of operations from APPLICATIONS 104. ENGINE 206 stores all execution data received from APPLICATIONS 104 in STORE 204. ADMINISTRATORS 108 can view the execution details via UI 202.

         We have discussed the implementation of the general model. Fig. 3 illustrates

15   a refined embodiment. In particular, the refinement is a mechanism of communication between ENGINE 206 and APPLICATIONS 104. When ENGINE 206 needs to pass data to an APPLICATION 104 it creates a *Job Order* containing the data and places it in STORE 204. The boxes labeled AGENT 1 (302a) through AGENT N (302n) represent components of our system that correspond to

20   APPLICATIONS 1 (104a) through N (104n). An AGENT 302 has the responsibility of picking up *Job Orders* for its APPLICATION 104 from STORE 204, passing the data contained in the *Job Order* to the APPLICATION 104, monitoring execution, and recording results of operations in STORE 204. After an AGENT 302 updates a *Job Order* with execution details it received from its APPLICATION 104, ENGINE

25   206 picks up the *Job Order*, determines whether operation performed by the APPLICATION 104 was successful, and continues working according to the rules defined in the *Action*.

         The AGENTS 302 communicate with APPLICATIONS 104 through an operating system. During *Action* creation DEVELOPER 106 specifies a special string

30   (*command*) for each APPLICATION 104. ENGINE 206 retrieves the *command*, makes substitutions of portions of the string for USER 110 data and stores the resulting *command* in *Job Order* together with USER 110 data. AGENT 102

retrieves the *command* from *Job Order* and presents it to operating system for execution.

It may not always be possible or convenient to record the whole operation to be performed by APPLICATION 104 in one *command* string. In such cases DEVELOPER 106 has the option of creating a custom executable, containing the full operation to be performed by the APPLICATION 104. Boxes labeled C1 (304a) through CN (304n) in Fig. 3 represent these custom executables. The *command* then becomes a simpler string that calls the custom executable. Although there is no restriction on the complexity of the custom executables, we expect them to usually be simple scripts. Development of the custom executables is simplified due to a number of factors. First of all, the AGENT 102 typically runs on the same machine as the custom executable and (usually) the APPLICATION 104, thus there is no need for communications over network. (Of course, in its broadest aspect, the invention is not so limited.) The custom executable can be written in any language of DEVELOPER's (106) choosing, and thus could be native to the APPLICATION 104 and the operating system environment. Finally, the AGENT 104 makes user data readily available to the custom executable. We support several ways of passing data: as part of *command* string, on standard input and via environment variables.

We now discuss an overview of a detailed implementation. We used Directory Server as STORE 204. Throughout this document we use the terms Directory Server, LDAP server and LDAPSVR interchangeably to mean Directory Server. Directory Servers are produced by many commercial and non-commercial organizations (e.g. Netscape Corp. and University of Michigan). Additional information about Directory Servers and the protocols used to communicate with them (LDAP and LDAPS) can be found in RFCs 1777 and 2251 at http://www.cis.ohio-state.edu/hypertext/information/rfc.html as well as documentation provided by manufacturers of Directory Servers. We used Netscape LDAP SDK library for communication with Directory Server. Documentation on LDAP SDK is available from Netscape. We used a collection of web pages and CGI scripts as UI 202. Users (DEVELOPER 106, ADMINISTRATOR 108 and USER 110) of the system can access UI 202 via a web browser (404, 406 and 408, respectively. The code for most components is written in C++. To compile the C++

code we used GNU egcs compiler produced by Free Software Foundation. Portions of code for UI are written in Perl5, Bourne Shell, HTML and C++.

Fig. 4 is a schematic representation of our detailed implementation. Boxes with solid borders represent physical machines. Boxes with double borders represent

5 software components developed as part of our system. Boxes with dotted border are either standard third party applications or pre-existing applications. Boxes labeled A1 through AN denote APPLICATION 1 (104a) through APPLICATION N (104n). Boxes with dashed borders are optional components developed by DEVELOPER 106. Arrows in the diagram represent connections (network or other). Arrows are drawn in

10 the direction in which the connection is initiated. Dotted arrows represent connections made via LDAP or LDAPS protocols. Dashed arrows represent connections made via HTTP or HTTPS protocols. Thus all network connections in our system are made via standard protocols with secure counterparts. Our implementation includes the following components: ENGINE 206, AGENT 102,

15 LIBRARY (which is common to every component as described in greater detail below) . and UI 202.

We will describe each component in more detail below, however, first let us turn to the data model we used in our implementation. We define the following classes of objects for storage: *CPAT, ParamW, User, Group, Engine, Agent, Action,*

20 *Job, Request, Job Order* and *Folder*. Each of the classes has a corresponding class defined in our C++ code and in LDAPSVR schema. Throughout this description, we use the same italicized object class names to also mean objects of the class as well as the general notions the objects are intended to represent. *CPAT* is the parent of all the other classes we store in LDAPSVR and has the standard class *top* as its parent. We

25 do not use objects belonging to class *CPAT* directly, but rather use the class as a virtual parent class. *CPAT* has a field *name* that is stored in attribute *cn*. We use *names* as user-friendly object identifiers that are suggestive of the object's function. We store all of our objects in the same tree in LDAPSVR. The top of our tree is a *Folder* called TOP. *Folders* are used to organize data inside of our tree in LDAPSVR

30 into subtrees. Users are given an interface to build arbitrary trees of *Folders*. Objects of classes *Engine, ParamW, Agent, Action* and *Job* can be stored anywhere inside of our tree. For convenience we impose additional structure on our tree. Directly

underneath *Folder* TOP, we create *Folders* Users, Groups, Collections and System.
Folders Users and Groups contain objects of type *Users* and *Groups* respectively.
Folder System contains system information. *Folder* Collections contains user
defined *Folders*, also called *Collections*. User defined objects go inside of

5   *Collections*.

See Fig. 5 for an example of a tree. In this figure boxes with solid border
denote *Folders* and boxes with dotted border denote other objects. An object of class
Engine (e.g., the object labeled 502) stores configuration information for an ENGINE
206. An object of class *Agent* (e.g., the objects labeled 504a and 504b) stores

10   configuration information for an AGENT·102. An object of class *Action* (e.g., the
objects labeled 506a and 506b) stores an *Action* definition as described above. *Action*
has a field *script* that holds a list of DNs of *Jobs* to be executed with some additional
syntax. (DN stands for Distinguished Name, a unique identifier of a record in an
LDAP database. See your Directory Server documentation or RFCs for more

15   information on DNs.)

Field *paramDN* of *Action* stores a list of DNs of *ParamWs*. An object of class
ParamW is a definition of a *parameter*, and contains information on how to present it
to USER, default and allowed values as well as syntax rules. An object of class *Job*
(e.g., the objects labeled 508a, 508b and 508c) stores data needed to interact with a

20   specific APPLICATION. *Job* has a field *param* that stores a list of *parameters*
needed to execute the *Job*, and a field *rval* that stores a list of *parameters* that the *Job*
will return. *Job* also has a pointer to the *Agent* that is to execute the *Job* and a field
command that stores the *command* as described above. *Job* has a field *notify* that
contains the email addresses of developers/maintainers of the *Job* who will be notified

25   if this *Job* fails. An object of class *Request* stores *Request* as described above. A
Request is an instance of an *Action* execution. *Requests* inherit *names* from
corresponding *Actions*. *Request* also contains a field *submitterDN*, which identifies
the USER who submitted the *Request*. An object of class *Job Order* stores *Job Order*
as described above. A *Job Order* is an instance of a *Job* execution. *Job Orders*

30   inherit *names* from *Jobs*. *Job Orders* also contains a field *start_time* that stores the
timestamp of beginning of execution. ENGINE 206 uses this field to monitor how
long the *Job Order* takes to complete.

Our implementation includes four components: ENGINE, AGENT, UI and LIBRARY.

LIBRARY compiles into libutil.a, a library that contains procedures used by code in other components. Source code for LIBRARY is located in directory
5   util.

Source code for ENGINE is located in directory engine. Below is an outline of the component. Please refer to the source code for details. ENGINE is intended to run as a daemon. It reads its configuration files and then proceeds to main loop. ServiceDN configuration parameter stores the DN of the tree in which ENGINE
10   works. This would normally be TOP. ENGINE searches in its working tree for objects of type *Action* and creates a list of all *Actions* that need to be serviced. It then services each *Action* on the list. After each *Action* has been serviced, ENGINE sleeps for a specified interval of time before proceeding to the next iteration of the main loop. ENGINE expects to find a specific tree structure underneath an *Action*. An
15   example of such structure is depicted in Fig. 6.

Each *Action* 602 has three *Folders* underneath it: In 604, Queue 606, and Out 608. *Folder* In contains new *Requests* posted by USER. *Folder* Queue is where *Requests* reside while being executed. *Folder* Out is for *Requests* that have been executed (completed or failed). To service an *Action*, ENGINE first processes its In
20   *Folder*. ENGINE reads the definition of the *Action* and all *Jobs* mentioned in the script. It then parses each *Request* and checks that USER supplied all necessary parameters. ENGINE moves the parsed *Request* into Queue and creates *Job Order* objects underneath it. ENGINE creates one *Job Order* per each *Job* mentioned in the script. *Job Orders* get *IDs* made up of the *Request ID* with *Job sequence number*
25   appended. *Job sequence numbers* come from numbering all *Job* references in the script in the order they appear. Each *Job Order* contains enough information for an AGENT to be able to execute it. It includes *command*, *Job* definition data and USER data from the *Request*.

Each *Request* has a *status* field that is used by ENGINE. When ENGINE first
30   puts the *Request* into Queue, it gives it *status* of HOLD to indicate that it has not completed parsing it yet. After all *Job Orders* are created underneath the *Request*,

ENGINE changes the *Request's status* to RUNNABLE. Each *Job Order* also has a field *status* that is used by ENGINE and AGENTS. ENGINE first creates a *Job Order* with a *status* of HOLD.

5

After processing *Folder* In of each *Action* on the list, ENGINE moves on to process *Folder* Queue. For each *Action* it retrieves all *Requests* in Queue and processes them one by one. With reference to Figs. 7 and 8, we describe what ENGINE does with each *Request*. We also refer to the source code in file engine/EngineD.cc (especially procedures EngineD::state_machine and

10

EngineD::wait_for_bg_jobs). Each *Request* has a field *pc* that holds the sequence number of the *Job Order* currently being executed. If *pc* is less than the total number of *Job Orders* in the *Request* (step 702), the ENGINE checks the *Job Order* pointed to by *pc* (see Fig. 7).

If the *Job Order* is a background *Job Order* (step 704), the ENGINE checks to see whether it has been placed (step 706). If the *Job Order* has not been placed, the

15

ENGINE places the *Job Order* (step 708) before proceeding. The ENGINE then increases *pc* by one (step 710) and moves on to the next *Job Order*.

If the *Job Order* is a foreground *Job Order*, the ENGINE checks to see whether it has been placed (step 712). If the *Job Order* has not been placed, the ENGINE places the *Job Order* (step 714) and moves on to work on other *Requests*. If

20

the *Job Order* has been placed before, the ENGINE checks the AGENT's queue to see if the *Job Order* has been completed (step 716). If not, the ENGINE moves on to the next *Request*. If yes, the ENGINE removes the *Job Order* from the AGENT's queue (step 718), determines whether the *Job Order* has succeeded or failed (step 720) and updates the *Job Order* record in the *Request's* subtree. If the *Job Order* was

25

successful, the ENGINE pulls return values from the *Job Order* (step 722) and stores them in the *Request*. The ENGINE then increases *pc* (step 710) and moves on to the next *Job Order*.

If the *Job Order* has failed or there are no more *Job Orders* to process for this *Request*, the ENGINE waits for all background *Job Orders* (see figure 8). To do this,

30

the ENGINE cycles through all background *Job Orders* (step 802) and removes completed ones from AGENTS' queues. If it encounters any background *Job Orders*

that have not yet finished execution (step 804), ENGINE moves on to process other *Requests*. If all background *Job Orders* have finished, ENGINE determines the *status* of the *Request* (step 806). If all *Job Orders* in the *Request* have completed successfully the *status* of the *Request* is COMPLETE, otherwise the *status* is ERROR.

5   Lastly, the ENGINE moves the whole *Request* subtree from *Folder* Queue to *Folder* Out (step 808) and goes on to do other work.

To place a *Job Order*, ENGINE first determines values of all *parameters* needed by the *Job Order*. ENGINE takes *Request's parameter* values and assigns them to the *Job Order parameters* taking into account *parameter mappings* defined in

10 the *script*. *Parameter mappings* allow a *Job Order parameter* value to be an arbitrary string with references to *Request's parameters*. For example, suppose *Request* has *parameter* x, *Job Order* has *parameter* y, and *parameter mappings* in *script* specify that y="%<x> number". Then if x has value of "telephone", y will get value of "telephone number". In this example %<x> is a reference to *parameter* x. If there is

15 no *parameter mapping* defined for a *parameter*, its value is set to be the value of identically named *Request parameter*. After ENGINE determines values of all *Job Order parameters*, it makes substitutions of *parameter* references for *Job Order parameter* values in *command* string and standard input data (*input*). Finally, ENGINE writes the *Job Order* in AGENT's queue with *status* of RUNNABLE.

20 To pull *return values* (*rvals*) from a *Job Order*, ENGINE consults *reverse parameter mapping* definitions specified in *script*. *Reverse parameter mappings* follow the same conventions as ordinary *parameter mappings* we described in the previous paragraph. *Reverse parameter mappings* define *Request's parameter* values via arbitrary strings with references to *Job Order return parameters*. If a *Request's*

25 *parameter* is not explicitly mentioned in *reverse parameter mappings* its value is not affected even if there is an identically named *Job Order return parameter*.

Source code for AGENT is located in directory agent. AGENT is intended to run as a daemon. It reads its configuration files and goes into main loop. ServiceDN configuration parameter tells AGENT where its record is in LDAPSVR. AGENT

30 expects ENGINE to place all *Job Orders* for AGENT right underneath AGENT's record in LDAPSVR. AGENT retrieves all *Job Orders* in its queue with *status* RUNNABLE. It then services the *Job Orders* one by one. To service a *Job Order*

AGENT forks off a child process (CHILD) and waits for it. CHILD prepares the
environment and executes the *command*. Standard input, standard output, and
standard error streams of CHILD are all connected to the AGENT. Besides the
standard streams we also create an RVALS stream for passing return values from
5   CHILD to AGENT. CHILD can access RVALS stream on file descriptor 3. The
format of return values is one name-value pair per line with equality sign separating
name from value. While waiting for CHILD to complete AGENT receives and
appends to *Job Order log* all messages written by CHILD to standard output and
standard error streams. AGENT also supplies *input* data to CHILD via the standard
10   input stream and receives return values via RVALS stream. AGENT has time
limitations on how long to let CHILD run. If CHILD does not exit on its own within
the specified time period, AGENT will first send it a SIGTERM and then a SIGKILL
signals causing CHILD to abort execution. No matter what caused CHILD to exit,
AGENT gets and parses CHILD's exit status and appends its findings to the *Job*
15   *Order log*. If CHILD exited with status 0, AGENT sets the *Job Order status* to
COMPLETE. Otherwise AGENT considers that the execution failed and sets the *Job*
Order status to ERROR. AGENT updates the *Job Order* record in LDAPSVR with the
new *status*, *log* and *rvals* received from CHILD. AGENT then moves on to service
the next *Job Order*.

20   UI includes of two subcomponents: CUI and CGIUI. The source code for CUI
is located in directory ui and compiles into four executables: get_obj, move_obj,
update_obj and run_action. The executables provide a low-level interface for
manipulating objects in LDAPSVR and posting *Requests*, and can be used to batch up
operations in a script or to perform operations from languages that support system
25   calls (e.g. C or Java). The get_obj executable retrieves objects from LDAPSVR and
prints them to standard output in URL-encoded form. The move_obj executable
moves an object in LDAPSVR or removes an object from LDAPSVR. The
executable takes an argument cmd that can have two values: del and move. If the
value of the argument is del, the executable deletes an object from LDAPSVR. If the
30   value of the argument is move, the executable moves an object in LDAPSVR. The
update_obj executable makes changes to an existing LDAPSVR object or creates a
new LDAPSVR object. The run_action executable posts new *Requests* in

14

LDAPSVR. It retrieves *Action* object from LDAPSVR and verifies that USER has supplied sufficient data for a *Request*. It then generates a new *Request* ID and creates a new *Request* object. It posts the new *Request* into In *Folder* in the *Action's* subtree in LDAPSVR. To ensure uniqueness of the *Request ID*, run_action constructs it out of a timestamp, machine ID and process ID. This construction also allows to search *Requests* based on the time of posting or what machine they were posted from. Run_action prints the new *Request* to standard output in URL-encoded form. In case an error is detected during execution, all four executables in CUI output the error on standard error stream.

CGIUI component is written in Perl and Bourne Shell. There are three Bourne Shell scripts: console, admin and edit_object. They are located in cgi-bin directory of the web server. All of these scripts are simple wrappers of identically named Perl scripts. Bourne Shell scripts are used to set up environment for the corresponding Perl scripts. Refer to source code for more details on the Bourne Shell scripts.

The Perl code for CGIUI is located in perl directory and consists of a Perl module CPAT.pm, its submodules and three Perl CGI scripts: edit_object, console and admin. The Perl modules are used by the Perl scripts. Perl modules also provide a convenient API to our system for developers writing in Perl.

The script console was made with the non-technical user (USER) in mind. Therefore the web pages it creates typically have little technical detail in them so as not to overwhelm USER. The console script is for executing *Actions* and monitoring their progress. The Main Page shows all *Actions* USER is authorized to run categorized by *collections*, and various ways for USER to check on existing *Requests*. When USER selects an *Action* from the list, based on the information stored in the *Action* and all of its *ParamWs* the script creates Run Action Page for USER. Also if DEVELOPER has specified an address of a custom Run Action Page in *formURL* attribute of the *Action*, the script will redirect BROWSER to the custom Run Action Page. The Run Action Page queries USER for all necessary parameters that are needed to execute the *Action*. When USER presses Run Action Button on the page the script does syntax checks on *parameter values* and assuming all is well attempts to post a new *Request* to LDAPSVR. Upon successful completion the script displays

Successful Completion Page letting USER know what the ID is for the new
Request. If USER attempts to submit Run Action Page with invalid values of
parameters, console displays the Run Action Page again, but with error message
at the top, letting USER know what needs to be corrected. If there were any errors
5   during submission of Run Action Page, the script displays Error Page letting
USER know what the error was and what parameters USER submitted. On the Main
Page, USER is also given the capability to search *Requests* based on portion of
Request ID, any *parameter* value or submission time. If USER uses the search
capabilities, the script searches LDAPSVR based on the search options selected by
10   USER and displays Search Results Page. Search Results Page displays results
of the search as a numbered list. Each list entry includes the object's *name, ID, status*
and DN. *Status* fields are color-coded so it is easy to see which *Requests* or *Job
Orders* have been completed, which ones are still running, and which ones have
failed. By pressing on the Number Button of each list entry, USER can get detailed
15   information about the *Request.* A detailed Request Page displays the *Request's* ID,
name, status, log, DN, *pc, parameters, subbmitterDN* and a numbered list of *Job
Orders* together with their *names* and *statuses.* USER can view Job Order Page by
pressing the Number on the Request Page. Job Order Page displays the *Job
Order's ID, name, status, log, parameters, start_time, notify, return values,* DN and
20   *Request's* DN. Job Order Page also displays a View Request Button that allows
USER to view Request Page of the parent *Request.* The Main Page gives USER
additional utility functionality, such as log out, browse help files, check user identity
and create custom reports.

       The script admin was made for DEVELOPERS and ADMINISTRATORS.
25   The Main Admin Page displayed by the script allows ADMINISTRATORS to
configure the application, create and manage users of the application, create and
manage groups, manage user and group privileges. Also, for debugging purposes,
admin gives a more advanced interface to browsing Requests and Job Orders. Main
Admin Page also links to edit_object script for direct interaction with objects
30   stored in LDAPSVR.

       The script edit_object was made with the advanced technical user in mind.
It would normally be used by DEVELOPERS and ADMINISTRATORS. Users can

16

view, create and modify objects using this script. The objects that can be manipulated
by this script are *Actions*, *Jobs*, *ParamWs*, *Agents*, *Engines* and *Folders*. The Front
Page allows multiple search options for retrieving objects that users would like to
edit. To create a new object the user has to specify a new Base DN that does not

5   conflict with any other DN in LDAPSVR. To create a new object, the user has to
press the New Button. To edit an existing object, the user has to press the Edit
Button. Whether user is creating a new object or editing an existing one, the page
that comes up is Edit Page. In case of a new object all fields in the Edit Page are
left blank. In case of an existing object, the fields are populated with values from

10   LDAPSVR. On the Edit Page, user can change the values of object's fields.
Generally user is provided with three buttons on an Edit Page: Commit Object
Button, Revert Object Button and Update View Button. Commit Object
Button writes the changes to LDAPSVR. Revert Object Button reverts the fields
to the values they have in LDAPSVR. Update View Button checks consistency of

15   the object and reports any problems to the user without writing to LDAPSVR or
erasing changes made by user. In case the object is an *Action*, user is provided with a
search capability for retrieving *Jobs* he would like to add to the *script*. User can also
change the order of *Jobs* in the *script*, specify how *Action parameters* are mapped to
Job parameters (via *parameter mappings*) and where the *Job's return values* would

20   be stored in the *Action parameters* (via *reverse parameter mappings*). Users are also
given the capability to search and insert *ParamW* objects into *Action* definition.

It is now discussed how to produce a particular implementation. Choose a
Solaris 2.6 SPARC machine to be your build machine. Copy the source code into an
identical directory structure on the build machine. Unpack Netscape LDAP SDK in

25   directory /share/Depot/ldapsdk-30-SOLARIS-export-ssl on the build
machine. Run make in the top-level directory (/share/Kiki/WF/prod). Use
GNU make version 3.75, GNU egcs compiler version 2.91.57 and Netscape LDAP
SDK version 3.0.

Install a Directory Server (Netscape, University of Michigan, or similar) on a

30   machine. Configure the Directory Server to recognize suffix of O=NONE. Add
schema rules to the Directory Server. Schema rules are contained in Schema
directory. File attributes contains our definitions of attributes. File classes
contains our definitions of object classes. Start the Directory Server. Add contents of

17

LDIF file Schema/first.ldif to the Directory Server database. See your
Directory Server documentation on how to add records in LDIF format.

On your build machine, edit global system configuration file
/share/Kiki/WF/prod/syscfg. Set the default bind DN (bdn) and password
5   (bpw) to be the DN and password of LDAP server user who has full control over
o=NONE subtree (e.g. Directory Manager). Set the LDAP server (server) to be the
hostname of the machine on which you installed Directory Server.

Choose a Solaris 2.6 SPARC machine to be your ENGINE machine. If this is
different from your build machine, copy the entire directory structure under
10   /share/Kiki/WF/prod and /share/Depot/ldapsdk-30-SOLARIS-
export-ssl from build machine to the target machine. Go to
/share/Kiki/WF/prod/engine on the target machine and touch a file log.
Make sure that LD_LIBRARY_PATH environment variable in your shell points to
/share/Depot/ldapsdk-30-SOLARIS-export-ssl/lib and is exported
15   to children processes.

Then run ./engined cfg=./cfg & to start the ENGINE.

Install and configure a Web server (Netscape Enterprise, Apache, or similar)
on a Solaris 2.6 SPARC machine. Install Perl 5.02 on the same machine. Make sure
that the Perl 5.02 executable is accessible as /usr/bin/perl. Copy directory
20   structure under /share/Kiki/WF/prod from build machine to the Web server
machine. Set up a cgi-bin directory for the Web server. Copy Bourne shell scripts
from /share/Kiki/WF/prod/cgi-bin to the cgi-bin directory and set
executable bits on. Start the Web server.

It is now described how to use the system discussed and described above.
25   There are two main uses of our system: developing in-house solutions and using the
solutions developed with the help of our system. To illustrate how one would use our
system, let us turn to an example. As we mentioned in the Background, in-house
solutions may implement business methods or improve operations. Let us focus on an
in-house solution that improves internal operations of a business by automating
30   incoming process. An "incoming process" is a process that a company follows
whenever a new employee is hired. If a company does not have its incoming process
automated, all the steps of the incoming process have to be carried out manually.
Consequently, it is costly to hire new employees because of the manual labor involved

in the incoming process. Moreover, manual incoming process results in costly delays. Typically, a new employee is unable to perform his or her duties for the first two weeks on the job because of the delays caused by manual incoming process. However, automating the incoming process is also a costly task because of the

5   number of different technologies involved and high complexity of the resulting solution. A typical company will spend one to two years automating the incoming process. Unfortunately, because of rapidly changing environments, an in-house solution that took two years to build will most likely be out of date by the time the company starts using it. We explain below how to automate the incoming process

10 using our system. If one follows our instructions, the automation should only take a few weeks.

A typical incoming process include updating an HR system with employee information, creating an email account for the new employee, issuing an electronic badge for identification and building access, setting up voice mail, ordering

15 equipment, installing software and many other steps. To simply the discussion here, let us assume that the incoming process includes only the first three steps. Thus, we have three APPLICATIONS that need to be updated with information about the new employee: HR system, Email system and Security system. Suppose HR system is located on machine A, Email system is located on machine B and Security system is

20 located on machine C. Let us assume that the hostname of the web server from the previous section is webserver.

Point your BROWSER to http://webserver/cgi-bin/edit_object. Choose to add a New Collection. Give the New Collection an ID of em and choose GO. Give it a name of Employee Management. You will create the rest of your objects in this

25 collection.

For each of the APPLICATIONS, we perform the same four steps. First, create an Agent. Second, install and start an AGENT. Third, write custom executable that updates the APPLICATION. Fourth, create a Job. Because of the similarity, we will only explain how to perform these steps for the first APPLICATION.

30 Point your BROWSER to http://webserver/cgi-bin/edit_object. Choose to Create New Object. Choose to create a new Agent in Employee Management collection with ID of HR_agent. Press

`Create Object Button`. On the next page enter *Agent* name to be HR `agent` and press `Commit Agent Button`. A new *Agent* has been created.

To install AGENT, copy entire directory structures under `/share/Kiki/WF/prod` and `/share/Depot/ldapsdk-30-SOLARIS-`

5   `export-ssl` from build machine of previous section to machine A. Go to `/share/Kiki/WF/prod/agent` on machine A and create file `log`. Also, edit file `cfg` and set the value of `servicedn` to be `objid=HR_agent`, `objid=em`, `objid=Collections`, `objid=TOP`, `o=NONE` (just like it showed on the Create Agent screen in `edit_object`). Make sure that

10   `LD_LIBRARY_PATH` environment variable in your shell points to `/share/Depot/ldapsdk-30-SOLARIS-export-ssl/lib` and is exported to children processes.

Start the AGENT by typing `./agentd cfg=./cfg &` .

The third step employs knowledge of the HR application and development

15   skills. Write a custom executable (script) that expects four command line arguments. The script should update the HR application with the following information about the new employee: first name, last name, social security number and department. The script should take the data about the new employee from command line arguments. Let us assume that the first argument is employee's first name, the second argument is

20   last name, the third argument is social security number and the fourth argument is department. Place the script in file `/scripts/hr_add` on machine A and set it's executable bit on. You may wish to test the script to make sure it performs the correct operation.

To create a *Job*, point your BROWSER to `http://webserver/cgi-`

25   `bin/edit_object`. `Choose Create New Object`. Choose to create a new *Job* in Employee Management Collection with ID of `HR_job` and press `Create Object Button`. Set *name* to be HR `Job`. Press the `Select Agent Button`. On the next page, select the HR `Agent`. Set *params* to be `FirstName`, `LastName`, `SSN`, `Department`. Set *command* to be `/scripts/hr_add`

30   `%<FirstName> %<LastName> %<SSN> %<Department>`. Press the `Commit Job Button`. A new *Job* has been created.

Now repeat the process for the other two APPLICATIONS. At the end, you should have created three *Jobs*: HR `Job`, Email `Job` and Security `Job`.

Now create *ParamW* objects for each of the *parameters*: FirstName, LastName, SSN and Department. To create *parameter* FirstName, point your browser to http://webserver/cgi-bin/edit_object. Select Create New Object. Choose to create a new *Parameter* in Employee Management *collection* with ID of FirstName and press Create Object Button. Set *parameter* name to be FirstName and press Commit Object Button. Repeat the process for the other *parameters*.

The final step is to create a new *Action*. Point your BROWSER to http://webserver/cgi-bin/edit_object. Choose Create New Object. Choose to create a new *Action* in Employee Management Collection with ID of new_hire and press Create Object Button.   On the next page, set *Action* Name to be New Hire, insert *parameters* FirstName, LastName, SSN, Department in the order in which you would like them to appear on the form. Insert the three *Jobs* you have just created and press Commit Action Button. The development process is over and the new solution is ready to be used.

To use the solution, point your BROWSER to http://webserver/cgi-bin/console. Click on New Hire business process. On the next page, enter the new employee data in each field. For example, enter John in the FirstName field, enter Smith in the LastName field, enter 555-5555 in the SSN field and enter Marketing in the Department field. Press the Submit Button. The new employee will be added to HR system, Security system and Email system. You can get updates on the execution of your *Request* by entering a portion of the *Request ID*, employees name or submission time on the http://webserver/cgi-bin/console page. This will give you detailed information about execution of the *Request* and its *Job Orders*. If there are any errors, they will appear in the *Request* or *Job Order logs* and the execution of the *Request* will be stopped at the first *Job Order* that fails.

Particular features are now described. First, the web interface for developers and administrators is described. Our web interface for developers and administrators is intuitive and easy-to-use. It presents users with easy-to-understand descriptions of *Action* and *Job* definitions aiding in visualization of solution architecture. This description of solution architecture automatically stays up-to-date. Our web interface forces developers to think in terms of high-level modules. It does not clutter display

21

with details of irrelevant components, but allows developers to zoom to a component to get additional details. Our web interface allows administrators to monitor execution of *Requests* and browse archives of prior *Requests* and *Job Orders*. Administrators can easily debug and troubleshoot problems with the help of our web interface.

The automatic USER interface for new solutions is now described. When developers create a solution using our system, it automatically creates USER web interface for the new solution. Thus, developers do not need to be skilled in web interface design or spend their time writing the extra component. USERS benefit from automatic USER interface because every solution created with our system comes with easy-to-use USER interface. Developers get many options allowing them to easily customize USER interface. With our system, developers get the benefits of reusability, since individual USER interface components can be reused from one solution to another. In case USERS need a completely custom USER interface, developers are given the option to externally develop such a USER interface and integrate it into any solution developed with our system. To specify an externally developed USER interface for an *Action*, simply refer *formURL* field of the *Action* to the address of the custom USER interface.

The Network Communications is now described. Our system provides communications between all components of the new solution. Thus, developers do not need to be skilled in network communications or implement network connectivity inside of their components. Therefore the complexity of each individual component is greatly reduced.

Secure communications are also offered. All network communications in our system can be easily switched into secure mode. Secure protocols are very difficult to design and implement because the slightest flaws could invalidate the security of the whole protocol. In order to provide sufficiently high degree of security, secure protocols have to be tested out by a large community over a long period of time. These resources are never available to in-house developers. Lack of skill, time and tests while designing solutions with secure communications often results in low quality security. We use standard secure protocols, LDAPS and HTTPS, for secure communications. These protocols have been created by on-going efforts of many skilled developers and extensively tested by a very large Internet community.

Consequently, in-house developers using our system do not need to be skilled in secure communications or spend time and effort on designing secure communications into the solution. To enable secure communications simply configure the web server and LDAP server to work in secure mode. This will automatically switch network
5   protocols from LDAP and HTTP to LDAPS and HTTPS. See Figure 5 for more details.

Fault detection is provided. Our system provides fault detection at the highest level of component integration. When a fault is detected execution is stopped at the first failing component. Therefore, faulty data is not passed on to the other
10 components. Developers who design their own components do not need to worry how a fault in their component will affect the rest of the solution. Therefore, code for custom components is simplified. The failed component is clearly marked in *Request* allowing developers and administrators to find the problem quickly.

Logging is also provided, which is particularly useful for debugging problems.
15 Our system logs its own actions and decisions and provides developers with a mechanism to write debugging information to logs. Our system automatically logs all errors and execution details provided by external components and our own software. Logs created during *Job Order* execution are stored in the *Job Order* record. Higher level details are stored in *Request* logs. General problems are logged in ENGINE and
20 AGENT logs. Developers can generate additional debugging information by writing to the standard error stream. Since logs are automatically generated, collected into a central location and displayed over a web interface, developers do not need to design additional logging mechanisms into their components. Thus component code is simplified while administrators and developers are sure to get good debugging
25 information for every solution built with our system.

The components of our system employ a stateless design. ENGINE and AGENTS do not rely on state information stored in memory and store it in LDAPSVR instead. Therefore, ENGINE or AGENT can be restarted without disrupting its normal function. This stateless architecture leads to a more stable
30 system allowing for easy integration with high availability technologies (see below). In addition, state information in LDAPSVR can be examined for debugging purposes.

Our system integrates with high availability technologies. Our system has been designed for easy integration with high availability technologies. ENGINE,

AGENTS and CUI do not depend on the LDAPSVR connection to be available at all times or in a continuous fashion. If the connection is lost they will reconnect automatically. Moreover, if administrators specify a list of LDAP servers in the configuration file, ENGINE, AGENTS and CUI will try all servers on the list until they establish connection to one of them. On the other hand, ENGINE and AGENTS themselves can be set up as highly available components. If a failover occurs, the new instance of the component will pick up right where the old one left off because of the stateless design.

The capability to pass "return values" is provided. Developers can pass data generated by their components to other components via *Job Order* return values (*rvals*). In order to pass *rvals*, developers need to specify *return parameters* in *Job* definition. Custom executables and APPLICATIONS can pass data to AGENTS by writing *return parameter* name-value pairs to special RVALS stream as described previously. In *Action* definition, developers can use *reverse parameter mappings* to specify where *rvals* should be stored. ENGINE pulls *rvals* from complete *Job Orders* and places them into *Request parameters*. These *parameters* can later be passed to other *Job Orders*.

To allow for greater flexibility and easier code reuse, we provide a *parameter* substitution feature. It allows *Job* (or *Job Order*) *parameters* to be arbitrary strings with references to *Action* (or *Request*) *parameters*. *Parameter* substitution is described above.

A background job capability is also provided. Suppose we have an *Action* with three *Jobs* and each *Job* takes 1 hour to execute. If the *Jobs* are executed in sequence, the whole *Action* will take three hours to complete. However, if there are no dependencies between *Jobs* we can run them in parallel. Then the *Action* would take only one hour to complete. To allow the *Jobs* to run in parallel, we provide the background *Job* option. After placing a background *Job Order*, ENGINE goes on to placing the next *Job Order* without waiting for the background *Job Order* to complete. Before marking the whole *Request* as complete, ENGINE waits for completion of all background *Job Orders*.

We allow developers and administrators to specify administrator email addresses in *Job* definitions. If ENGINE encounters a failed *Job Order*, or if the *Job*

Order takes too long to COMPLETE, ENGINE sends an email notification to the administrator of the *Job*. This way administrators are automatically notified about problems within their components. Thus, error notification allows for distributed administration, described in greater detail later.

5   We allow inclusion of configuration files, making it easier to organize configuration information. For example, generic configuration data such as LDAP server and port can be stored in the main configuration file. Configuration files for ENGINE, AGENTS and CUI can then include the main configuration file to get all the generic configuration parameters. For single-valued parameters, our system uses
10   the first value it finds. Therefore if the main configuration file is included at the very end, its default values can be overridden by values specified before the inclusion. For multi-valued parameters our system collects all values specified in all configuration files. Therefore, custom configuration files can add extra values to the ones specified in the main configuration file. We do not impose restrictions on the number or depth
15   of inclusions.

       Our system collects all USER data and all execution data in LDAPSVR in searchable format. We provide limited search capabilities in our web interface. However developers can use our CUI executable `get_obj` to make more general searches. The executable accepts a general `filter` parameter that follows the
20   standard LDAP conventions. Developers, administrators and management can gain vital information from reports created using custom searches. These reports can shed light on the use of in-house solutions and give dynamic enterprise statistics.

       For security or performance reasons, administrators may wish to run several ENGINES simultaneously. Multiple ENGINES can work with the same LDAP server
25   and post *Job Orders* to the same AGENTS. However, it is important that the ENGINES service disjoint subtrees. Note that *Actions* serviced by ENGINE have to be located in its service subtree while *Jobs* and *Agents* can be located anywhere in our tree.

       Our system allows developers to use two authentication mechanisms: web-
30   based authentication and LDAP-based authentication. No matter what authentication mechanism is used, we make provisions for storing user ID in *Request parameters* for

tracking and use by *Job Orders*. The first mechanism forces all users of our system to prove their identity to the web server before they can access our system. This mechanism can be initiated by configuring the web server to require authentication before web pages from CGIUI are served to the users. Your web server

5 documentation will explain how to perform such configuration. In this scenario, CGIUI will be passed the user ID by the web server. By default LDAP-based authentication is used. CGIUI queries all users for their user ID and password. CGIUI then attempts to assume the user's identity in communications with the LDAP server. The LDAP server performs authentication and if user ID and password do not

10 match, it will refuse the communication. LDAP users and groups can be created via our admin interface. You should configure the LDAP server to disallow anonymous access. Information on LDAP server configuration can be found in the documentation for your LDAP server.

Many solutions benefit from authentication information. However new

15 authentication methods are difficult to design and require rigorous testing. By providing developers with access to these two authentication mechanisms we eliminate the need for implementation of custom authentication methods for newly created solutions. Thus, development effort and expertise required for creation of solutions with authentication are greatly reduced. As a side comment, developers

20 may choose to use both authentication methods simultaneously forcing user identity tracking in both web and LDAP server logs.

Our system supports two authorization models based on the two authentication methods described above. The first one combines web and LDAP server authorization features while the second one is purely LDAP server based. In the first

25 model, authentication is performed by the web server as described above. Administrator creates several instances of CGIUI with distinct configuration files. Each configuration file specifies an identity to assume when dealing with LDAP server. Web server determines whether a particular user is authorized to access a particular instance of CGIUI. CGIUI then assumes the identity specified in its

30 configuration file. LDAP server determines what kind of operations the identity is authorized to perform with the LDAP Database. To use this authorization model, administrator has to create distinct identities in LDAP server and give them rights (see

the documentation for your LDAP server on how to do it). Administrator has to install multiple instances of CGIUI (in separate directories) and specify distinct LDAP server identities in their configuration files. Administrator has to configure the Web server to authorize only specific groups of users to access different CGIUI
5   components (see your Web server documentation for details).

The second model assumes that LDAP-based authentication is used. In this case LDAP server can perform authorization as well. Administrator has to disable anonymous access and set up different rights for different users or groups of users. LDAP server will then automatically perform authorization according to the rules
10   specified by administrator. See you LDAP server documentation for more details.

Many solutions will benefit from built-in authentication and authorization. Thus, we save development time by providing these mechanisms. Moreover, these mechanisms can be used to provide security during the development process itself. Note also, that we can simultaneously have subtrees in LDAPSVR where anonymous
15   access is allowed, subtrees where only authenticated access is allowed but no authorization is performed and subtrees where only authorized access is allowed. Thus, administrators can configure authentication and authorization to closely fit enterprise needs.

When a secure solution is desired, developers may make use of the security
20   features provided by our system: secure communications, authentication and authorization. In addition, our system has been designed with architectural security that developers can further exploit using firewalls and additional configuration. All communications between components of our system happen over LDAP (or LDAPS) and HTTP (or HTTPS) allowing all other network ports to be locked down with
25   firewalls. Moreover, all LDAP connections are opened in the direction of LDAP server (see Fig. 4). Thus, APPLICATION machines that potentially contain important enterprise data (marked A1 through AN in Fig. 4) can be put on secure subnets with no network ports open towards them. Moreover, LDAP server itself can be put onto a secure subnet. If insecure APPLICATIONS need to access LDAP
30   server, only the LDAP (or LDAPS) port needs to be open. Since APPLICATION servers are the ones with vital data, they need to be protected the most. That is why we designed our system so that there is no active party connecting to APPLICATION

servers and dictating them what to execute. Instead, our system makes use of AGENTS, which are located on APPLICATION servers. AGENTS pull data from LDAP server and execute code located on APPLICATION servers. In addition, we propose secure AGENT features described later in this Detailed Description. Secure

5   AGENT features ensure that even in case of break-in into LDAP server, intruder still can not dictate APPLICATION servers what to execute. With secure AGENT features, developers will be able to specify *command* in AGENT configuration file instead of LDAP server. In addition, AGENT will be able to pass *parameters* to CHILD via standard input stream instead of in *command* string. This way, the only

10 data AGENT will receive from LDAP server will be *parameter* values. Thus, intruder can only change *parameter* values. Since *parameter* values are no longer part of *command* string, intruder can not dictate what code AGENT will execute on APPLICATION server.

Our system is designed to enforce a highly modular architecture on the newly

15 created solutions. Specifically, the software is split into separate modules and communication interface between modules is fixed in advance. Each of the modules is self-contained except that it communicates with other modules over the pre-defined interface. Modularity allows software engineers to develop modules in parallel thus shortening the time it takes to complete the whole solution. During the maintenance

20 cycle, any module can be replaced with new code without the need to make modifications to other modules as long as the new module adheres to the old communication interface. Since modularity expedites development and simplifies maintenance of code, software engineers are taught to develop modular code.

Unfortunately, modularity has two drawbacks. First, modularity lengthens

25 design stage requiring to split the code into modules and to define a communication interface. Second, it takes more effort to write modules strictly adhering to the communication interface standard. As discussed in the Background section, in-house development teams are typically focused on short-term benefits. Since most benefits of modularity are realized long term during maintenance cycle, in-house solutions

30 often lack modularity.

Our system pre-defines modular architecture and communication interface thereby shortening the design stage. Since it also provides communications between

modules and many other features described in this section, code for each module (component) is greatly simplified. Therefore, our system makes development of modular solutions easier and faster than writing non-modular solutions. Moreover,

5   we build on enforced modularity to deliver even greater benefits to in-house developers and administrators. We discuss later how modularity enforced by our system allows for asynchronous development, distributed administration and component-wise quality assurance.

Besides modularity, the other postulate of software engineering is reusable code. In our system, components of one solution can be naturally reused in other

10   solutions. To reuse a *Job*, simply refer to it in another *Action's script*.

Another feature not yet discussed allows for passing standard input to CHILD. To allow AGENT to pass data to CHILD on standard input the following changes need to be made to our source code. Update LDAPSVR schema to define an extra attribute *input* as case-sensitive string. Update *Job* and *Job Order* schema classes to

15   include the extra attribute *input*. Update *Job* and *Job Order* class definitions in `util/obj.h`, `util/obj.cc`, `util/Job_Order.h` and `util/Job_Order.cc` to handle the additional field *input*. Update procedure `Req_Builder::build_jo_proto` in file `util/Req_Builder.cc` to copy `job.input` to `jo.input`. Update procedure `Req_Builder::prep_jo` in the same file to replace

20   markers (*parameter references*) in `jo.input`. Update procedures `Job_Run::build_child_input` and `Job_Run::init` in files `agent/Job_Run.cc` and `agent/Job_Run.h` to copy `jo.input` to `chld_input`. Update package `CPAT::Edit::Job` of CGIUI (located in `perl/CPAT/Edit/Job.pm`) to allow developers and administrators to specify value for field *input* of *Job*.

25   To make AGENTS more secure, some embodiments provide the option of specifying *command* in AGENT'S configuration file rather than in LDAPSVR. To implement this option, the following changes are made to the source code. Define a new configuration parameter `Secure_Agent_Cmd` in file `util/WF_const.h`. Update procedure `Job_Run::init` in file `agent/Job_Run.cc` to check whether a value of

30   `Secure_Agent_Cmd` exists and if so, copy it to command instead of `jo.command`.

Alternative implementations are now discussed. It is possible to implement our system without AGENTS as shown in figure 2. To do this one could utilize remote shell (rsh) for execution of *Jobs* on remote APPLICATION systems. Remote shell is standard on all UNIX platforms and is available for Windows NT, 95 and 98.

5   If secure communications are desired, secure shell (ssh) can be used in place of ordinary remote shell. In this scenario DEVELOPER, USER and ADMINISTRATOR will interact with UI component. UI component will store information in STORE. ENGINE will pick up information from STORE and remotely execute (via rsh or ssh) *Jobs* on APPLICATION systems as specified in

10   *Action* definition. There are two advantages to this alternative implementation. First, it would require less implementation effort on our side. Second, it would be easier to install because in many cases no installation would be required on the APPLICATION systems. There are also two drawbacks to this implementation. First, the resulting system is less stable because ENGINE depends on APPLICATION

15   systems being available for *Job* execution. Second, the resulting system is less secure because an active remote party (ENGINE) is given the power to execute arbitrary commands on APPLICATION systems that could potentially hold very sensitive data. Considering the above factors, we came to the conclusion that our existing system is superior to this alternative.

20   While we chose LDAP server as STORE in our implementation, other means could have been used in its place. These include but are not limited to file systems, databases and web servers. We chose LDAP server over these alternatives because it has greater capability to organize and search data than file systems and web servers. On the other hand it is fast and lightweight compared to relational databases. In

25   addition, LDAP servers provide good authentication and authorization mechanisms and a well-tested secure communications protocol (LDAPS). Moreover, transparent referrals make LDAP servers superior compared to databases and allow for sophisticated distribution of data over network and security zones.

Let us point out, that if remote shell mechanism is used instead of AGENTS,

30   file system is used as a STORE and UI component is built into ENGINE, the resulting system would be extremely lightweight and easy to implement. This alternative implementation would consist of a single executable for ENGINE component residing

on a single machine and would closely resemble the general system shown in figure 1. Despite of the attractive simplicity, we chose our implementation because we believe it delivers more value to in-house solution developers, administrators and users.

Finally, let us note, that our system could have been implemented in languages
5   other than C++. In fact, we wrote the first model of the system in Java, C and C++. We found that C++ delivered better performance than Java and allowed for better code organization and reuse than C. Also we expect that future versions of UI will be partially implemented in JavaScript and Java in addition to Perl and Bourne Shell.

We now discuss how the described embodiments address the issues discussed
10   in the Background. In particular, we introduce two notions: asynchronous development and distributed administration.

While commercial solutions are developed in controlled and synchronized fashion, it is our belief that in-house development is better accommodated by asynchronous development model. First, the initial development team only exists to
15   develop the first version of an in-house solution. The solution later evolves over time as it is modified and extended by other developers. As discussed in the Background, it should not be assumed that developers making extensions to an existing solution participated in its initial development. Second, even initial development team would benefit from asynchronous development model, because with all team members
20   having other responsibilities and priorities synchronization slows the pace of development.

Since in asynchronous development model, it cannot be assumed that communications between developers are possible, pre-defined architecture, modularity and documentation become extremely important. Pre-defined architecture
25   eliminates the need for synchronized design stage in the beginning of development process. It also provides a framework for future development and ensures that resulting solution will be easy to extend. Modularity allows developers to work on their components without affecting other components of solution. Finally, documentation ensures that every developer has a good overall understanding of the
30   solution.

Our system is well suited for asynchronous development, because it pre-defines architecture and enforces modularity. Documentation of the pre-defined architecture will be provided with our system. In addition, high-level description of solution is created by our web interface from *Action* and *Job* definitions. Since the

5   description is dynamically created, it stays up-to-date throughout the lifetime of solution.

An integral part of development process is Quality Assurance. If asynchronous development model is to produce quality results, extensions of solution should be thoroughly tested. Enforced modularity of our system allows Quality

10   Assurance engineers (QAs) to take solution apart and test it component by component. This component-wise Quality Assurance shortens the test cycle and narrows required expertise. In addition, QAs benefit from up-to-date documentation and pre-defined architecture. Thus, Quality Assurance will produce much better results with our system than without.

15   Beyond asynchronous development model, developers using our system benefit from code reuse and built-in features that narrow required expertise and save development effort. Let us note that today all the features provided by our software have to be designed and built by in-house developers in each in-house solution. In our experience, these features account for eighty to ninety percent of code for each in-

20   house solution. Since with our software developers only need to perform ten to twenty percent of the work, our software dramatically increases efficiency of in-house development. Moreover, since our software will enjoy the benefits of high exposure to technical audience, eighty to ninety percent of each in-house solution built with our system will receive these benefits as well. Therefore, our system increases quality

25   and reliability of in-house solutions. In addition, we provide in-house developers with an array of security options including secure communications, secure architecture, authentication and authorization mechanisms.

We now discuss the concept of "distributed administration" to support in-house solutions. As we discussed in the Background portion, support of in-house

30   solutions is performed by administrators on demand. Thus, in-house support enjoys even less control and synchronization than in-house development. Moreover, due to complexity of in-house solutions, it is difficult to find administrators with all required

expertise. Based on these observations, we propose a distributed administration model. In this model, we do not assume that administrators are proactive, knowledgeable about the whole solution or work as a team to resolve problems. Instead, our system monitors execution, detects faults and notifies the right person

5   when problems arise. Thus, distributed administration ensures that problems are detected and debugged quickly. In our system, distributed administration is made possible by enforced modularity, fault detection and error notification. Pre-defined architecture and up-to-date documentation allow administrators to gain overall knowledge of solution. Beyond the overall understanding, an administrator only

10   needs to have narrow in-depth knowledge of the one component he supports, because of distributed administration. Thus, administrators are far less likely to introduce new problems by patches to existing in-house solutions. In addition, administrators can use our web interface to disconnect faulty components from an in-house solution with a few clicks of the mouse allowing the rest of the solution to function immediately.

CONCLUSION

15   To summarize, our invention enables enterprises to efficiently build and maintain high-quality in-house solutions that are secure and reliable, and that dynamically adjust to enterprise's needs.

APPENDIX A
SOURCE CODE LISTING

```
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/Makefile XXXXXXXXXX
# $Id: Makefile,v 1.17 1999/02/13 19:57:52 root Exp rt $

export PREFIX=/share/Kiki/WF/prod
export CXX=/opt/agcs/bin/g++
export AR=/opt/gnu/bin/ar
LDAP_LIB = -L/share/Depot/ldapsdk-30-SOLARIS-export-ssl/lib -lldapssl30
NET_LIB = -lxnet
WF_LIB = -L$(PREFIX)/util -lwf
export LDFLAGS = $(LDAP_LIB) $(NET_LIB)
export LOADLIBES = $(WF_LIB)
LDAP_INC = -I /share/Depot/ldapsdk-30-SOLARIS-export-ssl/include
export INCLUDES = -I $(PREFIX) -I $(PREFIX)/util $(LDAP_INC)
export CPPFLAGS = $(INCLUDES) -g
export BINDIR = $(PREFIX)/bin
export LIBDIR = $(PREFIX)/lib

# Main targets
all: cleanup util_dir engine_dir agent_dir ui_dir
release: rel_util rel_ui rel_engine rel_agent

# Recurse to subdirs
util_dir:
          cd util; $(MAKE)

ui_dir:
          cd ui; $(MAKE)

engine_dir:
          cd engine; $(MAKE)

agent_dir: util_dir
          cd agent; $(MAKE)

# Release
rel_util:
          cd util; $(MAKE) release
rel_ui:
          cd ui; $(MAKE) release
rel_engine:
          cd engine; $(MAKE) release
rel_agent:
          cd agent; $(MAKE) release

# Redo Targets
redo_ui:
redo:  clean all
          cd ui; $(MAKE) clean all
redo_util:
          cd util; $(MAKE) clean all
redo_engine:
          cd engine; $(MAKE) clean all
redo_agent:
          cd agent; $(MAKE) clean all

# Cleanup targets
cleanup:
          rm -f *~ core

clean: cleanup
          cd util; $(MAKE) clean
          cd ui; $(MAKE) clean
          cd agent; $(MAKE) clean
          cd engine; $(MAKE) clean

XXXXXXXXXX END /share/Kiki/WF/prod/Makefile XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/Schema/attributes XXXXXXXXXX
attribute actiondn           actiondn-oid         dn
attribute agentdn            agentdn-oid          dn
attribute command            command-oid          ces
attribute formurl            formurl-oid          ces
attribute jobdn              jobdn-oid            dn
attribute jobstates          jobstates-oid        ces
attribute log                log-oid              ces
attribute objid              objid-oid            cis
attribute param              param-oid            ces
attribute pc                 pc-oid               int
attribute rval               rval-oid             ces
attribute script             script-oid           ces
attribute status             status-oid           int
XXXXXXXXXX END /share/Kiki/WF/prod/Schema/attributes XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/Schema/classes XXXXXXXXXX
objectclass cpat
          oid cpat-oid
          superior top
          requires
                    objid
          allows
                    cn

objectclass job
          oid job-oid
          superior cpat
          requires
                    command,
                    rval,
                    param,
                    agentdn
```

APPENDIX A
SOURCE CODE LISTING

```
objectclass folder
        oid folder-oid
        superior cpat

objectclass engine
        oid engine-oid
        superior cpat
        allows
                actiondn,
                userpassword

objectclass agent
        oid agent-oid
        superior cpat
        allows
                host,
                userpassword

objectclass action
        oid action-oid
        superior cpat
        requires
                param,
                script
        allows
                formurl

objectclass request
        oid request-oid
        superior cpat
        requires
                actiondn,
                status,
                param,
                log,
                pc,
                script,
                jobstates

objectclass joborder
        oid joborder-oid
        superior cpat
        requires
                agentdn,
                status,
                command,
                log,
                actiondn,
                jobdn
        allows
                rval,
                param

XXXXXXXXXX END /share/Kiki/WF/prod/Schema/classes XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/Schema/first.ldif XXXXXXXXXX
dn: o=NONE
objectclass: top
objectclass: organization
o: NONE

dn: objid=TOP, o=NONE
objectclass: top
objectclass: cpat
objectclass: folder
objid: TOP
cn: Top of our tree

XXXXXXXXXX END /share/Kiki/WF/prod/Schema/first.ldif XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/syscfg XXXXXXXXXX
# Global system configuration parameters

# LDAP server
server=localhost

# Default bind DN and password
bdn=cn=Directory Manager
bpw=yourpassword

XXXXXXXXXX END /share/Kiki/WF/prod/syscfg XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/agent/AgentD.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/*****************************************************************
 *
 *    $Id: AgentD.cc,v 1.24 1999/02/18 02:53:33 rt Exp $
 *
 *    Desc: Main code for agent
 *
 ****************************************************************/

#include "WF.h"
#include "AgentD.h"
#include "Job_Run.h"

// main code for agent
void AgentD::do_work()
{
  try {
```

APPENDIX A
SOURCE CODE LISTING

```
    // set up logging
    log_setup();

    // get all we need to do the work
    check_params();
    build_ldap_params();

    // main work loop
    while ( true ) {
      // search incoming folder for job orders
      LDAP_Entry_Vec rv( ldap.search( inc_dn, LDAP_SCOPE_ONELEVEL, filter ) );

      // process job orders
      for (int i=0; i< rv.size(); i++)
        process_entry( rv[i] );

      // they asked us to go through queue only once
      if ( once )
        {
          write_to_log( "One pass through queue completed" );
          break;
        }

      // delay before reconnecting to ldap
      ldap.disconnect();
      sleep( interval );
    }
  }
  catch ( x_base x ) {
    write_to_log( (string)"Fatal error: " + x.msg );
    throw;
  }
  catch ( std::exception &x ) {
    write_to_log( (string)"Fatal error (stdlib): " + x.what() );
    throw;
  }
  catch ( ... ) {
    write_to_log( "Unknown fatal error." );
    throw;
  }
}

// cleanup some datastructures
AgentD::~AgentD()
{
  write_to_log( "Exiting ..." );

  if ( log_file && log_file != &cerr )
    delete log_file;
}

void AgentD::build_ldap_params()
{
  // build incoming folder dn
  inc_dn = config.val(ServiceDN_Param);

  // build search filter
  filter = (string)"(&(objectclass=" + Job_Order_Obj +
    ")(" + Status_Attr + "=" + num2str( Runnable_Status ) + "))";

  // set LDAP preferences
  ldap.prefs( config.server, config.bindDN, config.bindPW, interval );
}

// Make sure we have all params we need and figure out options
void AgentD::check_params()
{
  // Require these minimum parameters
  if ( config.server.empty() )
    throw( x_base( TPFX + "No LDAP server specified" ) );

  if ( config.bindDN.empty() )
    throw( x_base( TPFX + "No bind DN specified" ) );

  if ( config.bindPW.empty() )
    throw( x_base( TPFX + "No bind password specified" ) );

  if ( !config.has_a_val(ServiceDN_Param) )
    throw(x_base(TPFX+"No service DN specified"));

  // OPTIONS
  // Interval to sleep between connections to LDAP
  if ( config.has_a_val( Interval_Param ) )
    interval = safe_atoi( config.val( Interval_Param ) );

  // Go through queue only once
  if ( config.peek( Once_Param ) )
    once = true;
}

void AgentD::log_setup()
{
  // are we given a log file?
  if ( config.has_a_val( Log_File_Param ) )
    {
      string log_name = config.val( Log_File_Param );

      // open log
      log_file = new std::ofstream( log_name.c_str(), ios::app );
```

36

APPENDIX A
SOURCE CODE LISTING

```cpp
    if ( !(*log_file) )
        throw( x_base( TPFX + "Unable to open config file " + log_name ) );
    }

// write an opening message
write_to_log( "Agent starting ... " );
}


// Process a job order
void AgentD::process_entry( LDAP_Entry &e )
{
// log start of new job order
write_to_log( (string)"Processing job order " + e.dn );

try {
    Job_Order jo;
    jo.init_from_entry( e );
    Job_Run jr;

    // job run should only throw exceptions it
    // can not handle
    jr.run( jo );

    // update status and log of job order in LDAP
    jo.update_in_ldap( ldap );
}
catch ( x_base x ) { write_to_log( x.msg ); return; }
}

XXXXXXXXXX END /share/Kiki/WF/prod/agent/AgentD.cc XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/agent/Job_Run.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/*****************************************************************************
 *
 *    $Id: Job_Run.cc,v 1.25 1999/02/18 03:51:19 rt Exp rt $
 *
 *    Desc: Set up and run a job (called by agent)
 *
 *****************************************************************************/

#include "WF.h"
#include "Job_Run.h"

// for waitpid
#include <sys/wait.h>

// for SIGTERM and SIGKILL
#include <signal.h>

// for close
#include <unistd.h>

Job_Run::~Job_Run()
{
    // make sure pipes are closed
    for (int i=0; i<2; i++) {
        if ( in[i] >= 0 )
            close( in[i] );
        if ( out[i] >= 0 )
            close( out[i] );
        if ( rvals[i] >= 0 )
            close( rvals[i] );
    }
}

// does all the work
void Job_Run::run( Job_Order &jo )
{
    // figure out some params and setup
    init( jo );

    // were not able to figure out command to run
    if ( command.empty() )
        return;

    // execute the command
    execute_cmd();

    // finally write changes to job order record
    update_job_order( jo );
}

void Job_Run::update_job_order( Job_Order &jo ) {
    // assemble logs and figure out exit status
    prep_log_and_stat();

    // get return values
    if ( status == Complete_Status )
        copy_rvals( jo );

    // update status and log of job order
    jo.status = status;
    jo.log += log1.content();
}

void Job_Run::copy_rvals( Job_Order &jo ) {
    Param rmap;
    String_Vector lines = split( r_str, '\n' );
```

37

APPENDIX A
SOURCE CODE LISTING

```
for(int i=0; i<lines.size(); i++) {
    if ( !lines[i].empty() )
        log1.parent_write("Returned: "+lines[i]);
}

rmap.parse_str(r_str);
for (Param::iterator i=jo.r_map.begin(); i!=jo.r_map.end(); i++)
{
    string &name = i->first;
    if ( !rmap.peek( name ) ) {
        log1.parent_write("Child did not return value for ["+name+"]");
        status = Error_Status;
        break;
    }
    i->second = rmap[name];
}
}

void Job_Run::prep_log_and_stat()
{
    // assemble logs
    log1.append( log2.content() );
    log1.append( log3.content() );

    // figure out status
    if ( ex_stat >= 0 )
    {
        // child used exit(...)
        if ( WIFEXITED( ex_stat ) )
        {
            // job completed successfully
            if ( WEXITSTATUS( ex_stat ) == 0 )
                status = Complete_Status;

            log1.parent_write( "Child exited with status "
                              + num2str( WEXITSTATUS( ex_stat ) ) );
        }

        // child was terminated by signal
        if ( WIFSIGNALED( ex_stat ) )
        {
            log1.parent_write( "Child was terminated by signal "
                              + num2str( WTERMSIG( ex_stat ) ) );

            // was coredump made?
            if ( WCOREDUMP( ex_stat ) )
                log1.parent_write( "Core dumped." );
        }
    }
}

void Job_Run::init( Job_Order &jo )
{
    // initialize pipe fd's to -1
    in[0] = in[1] = out[0] = out[1] = rvals[0] = rvals[1] = -1;

    // figure out log size
    if ( config.has_a_val( Log_Size_Param ) )
        log_size = safe_atoi( config.val( Log_Size_Param ) );
    else
        log_size = Log_Size_Default;

    // set up logs
    log1.set_size( (int)(log_size * 0.8) );
    log2.set_size( (int)(log_size * 0.1) );
    log3.set_size( (int)(log_size * 0.1) );

    // figure out timeout interval
    if ( config.has_a_val( Job_Timeout_Param ) )
        timeout = safe_atoi( config.val( Job_Timeout_Param ) );
    else
        timeout = Job_Timeout_Default;

    // figure out command
    if ( jo.command.empty() )
    {
        log1.parent_write( "No command specified!" );
        return;
    }
    command = jo.command;

    // figure out child input
    build_child_input();
}

void Job_Run::build_child_input()
{
}

void Job_Run::spawn_off_child()
{
    // create pipes for communication with child
    if ( pipe( out ) != 0 )
        throw( x_sys( TPFX + "Can not create pipe: " ) );

    if ( pipe( in ) != 0 )
        throw( x_sys( TPFX + "Can not create pipe: " ) );

    if ( pipe( rvals ) != 0 )
        throw( x_sys( TPFX + "Can not create pipe: " ) );
```

38

APPENDIX A
SOURCE CODE LISTING

```
    // fork off the child
    child = fork();

    // child_code does not return
    if ( child == 0 )
      child_code();

    // Something is wrong
    if ( child == -1 )
      throw( x_sys( TPFX + "Can not fork: " ) );

    // close extra ends of pipes
    close( in[0] );
    close( out[1] );
    close( rvals[1] );

    // attach buffers to pipes
    ch_in.attach( in[1] );
    ch_out.attach( out[0], log_size );
    ch_rvals.attach( rvals[0], log_size );
    ch_in.put_buf( chld_input );
}

void Job_Run::execute_cmd()
{
    // opening log message
    log1.parent_write( "About to execute command: " + command );

    // setup and fork
    spawn_off_child();

    // let child do the work
    if ( read_write_wait( log1, now() + timeout ) )
      return;

    // timeout interval was exceeded - start sending signals

    // send SIGTERM
    log2.parent_write( "Timeout interval exceeded - sending SIGTERM to child" );

    if ( sigsend( P_PID, child, SIGTERM ) != 0 )
      throw( x_sys( TPFX + "While sending SIGTERM to child: " ) );

    if ( read_write_wait( log2, now() + 10 ) )
      return;

    // send SIGKILL
    log3.parent_write( "Sending SIGKILL to child" );

    if ( sigsend( P_PID, child, SIGKILL ) != 0 )
      throw( x_sys( TPFX + "While sending SIGKILL to child: " ) );

    if ( read_write_wait( log3, now() + 10 ) )
      return;

    // child must be stuck in kernel mode
    log3.parent_write( "Child did not respond to SIGKILL - giving up." );
    return;
}

bool Job_Run::wait_for_child( time_t max_time )
{
    int w;

    // wait for child to complete
    while( (w = waitpid( child, &ex_stat, WNOHANG )) == 0 )
      if ( now() > max_time )
        break;
      else
        sleep( 1 );

    // something bad happened
    if ( w < 0 )
      throw( x_sys( TPFX + "While wating for child: " ) );

    // true if child exited
    return ( w == child );
}

bool Job_Run::read_write_wait( Job_Log &a_log, time_t max_time )
{
    while( now() < max_time && !(ch_out.eof() && ch_in.finished()) )
    {
        write_to_chld();

        if ( !ch_out.eof() )
            chld_out_to_log( a_log );

        if ( !ch_rvals.eof() )
            read_rvals();

        sleep( 1 );
    }

    return wait_for_child( max_time );
}

void Job_Run::write_to_chld()
{
```

APPENDIX A
SOURCE CODE LISTING

```
if ( ch_in.finished() )
  {
    ch_in.close();
    return;
  }

ch_in.write_some();

if ( ch_in.finished() )
  ch_in.close();
}

void Job_Run::read_rvals() {
  // read as much as we can
  char *buf;
  int size = ch_rvals.read_some( &buf );

  // append to prev read rvals
  r_str.append( buf, size );

  // earase from buffer
  ch_rvals.get( size );
}

void Job_Run::chld_out_to_log( Job_Log &a_log )
{
  // read as much as we can
  char *buf;
  int r = ch_out.read_some( &buf );

  // beginning of line
  int beg = 0;

  // go through read char's and write each complete line to log
  for( int i=0; i < r; i++ )
  {
    if ( buf[i] != '\n' )
      continue;

    string tmp( buf + beg, i - beg );
    a_log.child_write( tmp );
    beg = i+1;
  }

  // some char's left without \n at the end
  if ( beg < r )
  {
    // pipe has been closed or is full - write out to log anyways
    if ( ch_out.eof() || ch_out.full() )
    {
      string tmp( buf, beg, r );
      a_log.child_write( tmp );
      beg = r;
    }
  }

  // earase from buffer everthing up to beg of next line
  ch_out.get( beg );
}

//==============================
// Job_Log class implementation
//==============================
void Job_Log::write( string e_line, bool from_child )
{
  // child log is too full for the whole line
  if ( from_child && log.size() + a_line.size() > max_size )
  {
    // this is not the first time
    if ( ! skipped )
      return;

    // attempt to write at least part of the line
    if ( log.size() < max_size )
    {
      string tmp;
      tmp.assign( e_line, 0, max_size - log.size() );
      log += time_stamp() + " CHLD " + tmp + '\n';
    }

    // let users know content is lost
    skipped = true;
    a_line = " ............ Log size exceeded - content lost ........... ";
    from_child = false;
  }

  // write to log - timestamp
  log += time_stamp();

  // who is the line from ?
  if ( from_child )
    log += " CHLD ";
  else
    log += " AGNT ";

  // content of the line
  log += e_line + '\n';
}

XXXXXXXXX END /share/Kiki/WF/prod/agent/Job_Run.cc XXXXXXXXXX
```

APPENDIX A
SOURCE CODE LISTING

```cpp
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/agent/Pipe_IO.cc XXXXXXXXXX
/*--- Mode: C++; -*-*/

/*****************************************************************************
 *
 *      $Id: Pipe_IO.cc,v 1.12 1998/12/16 00:16:29 rt Exp $
 *
 *      Desc: IO on pipes used to talk between agent and child
 *
 *****************************************************************************/

// for fcntl and O_NDELAY
#include <fcntl.h>

// for SIGPIPE and sigignore
#include <signal.h>

// for write, read, close
#include <unistd.h>

// for errno
#include <errno.h>

// our library
#include "WF.h"

// header for this file
#include "Pipe_IO.h"

void Pipe_Out::attach( int a_fd )
{
    if ( fd >= 0 )
        throw( x_base( TPFX + "Reattaching to a different file desc is not allowed." ) );

    fd = a_fd;
    out_eof = false;
    closed = false;

    // set O_NDELAY on fd
    if ( fcntl( fd, F_SETFL, O_NDELAY ) < 0 )
        throw( x_sys( TPFX + "fcntl: " ) );

    // set SIGPIPE to SIG_IGN
    if ( sigignore( SIGPIPE ) < 0 )
        throw( x_sys( TPFX + "sigignore: " ) );
}

void Pipe_Out::put_buf( string &s )
{
    // no new content provided
    if ( s.empty() )
        return;

    string tmp;

    // we still have some content left in buffer
    if ( out_buf && out_i < out_size )
        tmp = ( out_buf + out_i );

    // append new string to old content
    tmp += s;
    char *old_buf = out_buf;

    // allocate new buffer
    out_buf = dup_c_str( tmp.c_str() );
    out_size = strlen( out_buf );
    out_i = 0;

    // get rid of old buffer
    delete []old_buf;
}

void Pipe_Out::write_some()
{
    // nothing to write
    if ( !out_buf )
        return;

    int written;

    written = write( fd, out_buf + out_i, out_size - out_i );
    cout << "write_some: wrote " << written << " bytes" << endl;

    // error happened
    if ( written < 0 )
    {
        // pipe was closed by other party
        if ( errno == EPIPE )
        {
            out_eof = true;
            return;
        }
        else
            // strange error
            throw( x_sys( TPFX + "write: " ) );
    }

    // normal write
    out_i += written;
```

APPENDIX A
SOURCE CODE LISTING

```
)

//========
// Pipe_In
//========
void Pipe_In::attach( int a_fd, int a_size )
{
    // check that we are only called once
    if ( fd >= 0 || in_buf )
        throw( x_base( TPFX + "Attaching to fd is allowed only once." ) );

    // set up vars
    fd = a_fd;
    in_eof = false;
    max_size = a_size;

    // set O_NONBLOCK on fd
    if ( fcntl( fd, F_SETFL, O_NONBLOCK ) < 0 )
        throw( x_sys( TPFX + "fcntl: " ) );

    // allocate space for buffer
    buf_size = max_size * 2;
    in_buf = new char[ buf_size ];
}

int Pipe_In::read_some( char **head_ptr )
{
    int to_read = max_size - size;

    // got too close to the end of buffer - move to the beginning
    if ( end + to_read > buf_size )
        move_buf();

    // read from fd
    int r = read( fd, in_buf + end, to_read );
    *head_ptr = in_buf + beg;

    if ( r < 0 )
    {
        // some strange error
        if ( errno != EAGAIN )
            throw( x_sys( TPFX + "read: " ) );

        // empty pipe - nothing new read
        return size;
    }

    // other end of pipe is closed and pipe is empty
    if ( r == 0 )
    {
        in_eof = true;
        return size;
    }

    // normal read
    size += r;
    end += r;
    return size;
}

void Pipe_In::move_buf()
{
    // should not happen
    if ( size > beg )
        throw( x_base( TPFX + "Content of in buffer grew too big unexpectedly." ) );

    // copy buffer content
    int i, j;
    for( i=0, j=beg; i < size ; i++, j++ )
        in_buf[i] = in_buf[j];

    // adjust counters
    beg = 0;
    end = size;
}

void Pipe_In::unget( int some_num )
{
    // sanity checks
    if ( some_num > beg )
        throw( x_base( TPFX + "Asked to unget more than there is available." ) );
    if ( some_num < 0 )
        throw( x_base( TPFX + "Asked to unget a negative ammount." ) );
    if ( size + some_num > max_size )
        throw( x_base( TPFX + "Asked to unget too much." ) );

    // adjust counters
    beg -= some_num;
    size += some_num;
}

void Pipe_In::get( int some_num )
{
    // sanity checks
    if ( some_num > size )
        throw( x_base( TPFX + "Asked to get more than is available." ) );
    if ( some_num < 0 )
        throw( x_base( TPFX + "Asked to get a negative ammount." ) );
```

42

APPENDIX A
SOURCE CODE LISTING

```
// adjust counters
beg += some_num;
size -= some_num;
}
XXXXXXXXX END /share/Kiki/WF/prod/agent/Pipe_IO.cc XXXXXXXXX
XXXXXXXXX BEGIN /share/Kiki/WF/prod/agent/agentd.cc XXXXXXXXX
/*********************************************************
 *
 *   $Id: agent.cc.v 1.32 1999/02/03 09:59:02 rt Exp $
 *
 *********************************************************/
#include "WF_ext.h"
#include "AgentD.h"

int main (int argc, char **argv)
{
  try {
    config.init( argc, argv );
    AgentD agent;
    agent.do_work();
  }
  catch( x_base x ) { die( x.msg ); }
  catch( std::exception &x ) { die( x.what() ); }
  catch( ... ) { die( "Unknown fatal error." ); }

  exit( 0 );
}

XXXXXXXXX END /share/Kiki/WF/prod/agent/agentd.cc XXXXXXXXX
XXXXXXXXX BEGIN /share/Kiki/WF/prod/agent/child.cc XXXXXXXXX
/*-*- Mode: C++; -*-*/

/*********************************************************
 *
 *   $Id: child.cc,v 1.7 1999/02/18 03:20:18 rt Exp $
 *
 *   Desc: child_code run by child after it has been spawn
 *         setup pipes, file desc, env, and exec command
 *
 *********************************************************/
#include "WF.h"
#include "Job_Run.h"

// for perror
#include <stdio.h>

// for getrlimit
#include <sys/resource.h>

void Job_Run::child_code()
{
  // close extra ends of pipes
  close( out[0] );
  close( in[1] );
  close( rvals[0] );

  // map stderr/stdout/stdin to pipes
  if ( dup2( out[1], 1 ) < 0 )
    _exit( -1 );

  if ( dup2( out[1], 2 ) < 0 )
    _exit( -1 );

  if ( dup2( in[0], 0 ) < 0 )
    _exit( -1 );

  if ( dup2( rvals[1], 3 ) < 0 )
    _exit( -1 );

  // close all fd's above 3
  struct rlimit rl;
  if ( getrlimit( RLIMIT_NOFILE, &rl ) != 0 )
    _exit( -1 );

  for ( int i=4; i < rl.rlim_cur; i++ )
    (void)close( i );

  // create environment
  // pass NULL for now

  // finally execute the command
  execle( "/usr/bin/sh", "/usr/bin/sh", "-c",
          command.c_str(), NULL, NULL );

  // if exec was successful we should not get here
  perror( "Exec error" );

  exit( -1 );
}
XXXXXXXXX END /share/Kiki/WF/prod/agent/child.cc XXXXXXXXX
XXXXXXXXX BEGIN /share/Kiki/WF/prod/agent/AgentD.h XXXXXXXXX
/*-*- Mode: C++; tab-width: 4; indent-tabs-mode: nil; c-basic-offset: 4 -*-*/

/*********************************************************
 *
 *   $Id: Agent.h,v 1.6 1999/02/03 09:59:48 rt Exp $
 *
 *   Desc: Main code for agent
```

APPENDIX A
SOURCE CODE LISTING

```
.
.*********************************************************************/

class AgentD
{
public:
    AgentD()
        : interval( 5 ), log_file( &cerr ), once( false ) {}
    ~AgentD();

    // Main code for agent
    void do_work();

protected:
    LDAP_Wrap ldap;
    int interval;
    bool once;
    string filter;
    string inc_dn;
    ostream *log_file;

    void process_entry( LDAP_Entry &e );
    void build_ldap_params();
    void check_params();
    void log_setup();
    void write_to_log( string msg )
        { (*log_file) << time_stamp() << " " << msg << endl << flush; }
};
XXXXXXXXXX END /share/kiki/WF/prod/agent/AgentD.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/kiki/WF/prod/agent/Job_Run.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/*********************************************************************
 *
 *      $Id: Job_Run.h,v 1.19 1999/02/16 03:52:37 rt Exp $
 *
 *      Desc: Set up and run a job (called by agent)
 *
 *********************************************************************/
#include <unistd.h>
#include "Pipe_IO.h"

class Job_Log
{
public:
    // default constructor
    // however before using need to set_size
    Job_Log()
        : max_size( 0 ), skipped( false ) {}
    ~Job_Log(){}

    // two types of write's
    void child_write( string a_line ) { write( a_line, true ); }
    void parent_write( string a_line ) { write( a_line ); }

    // retrieve log's content
    string &content() { return log; }

    // add another log at the end (should be in the same format)
    void append( const string &str ) { log += str; }

    // set size to be accepted from child
    // (call this one before using log)
    void set_size( int size ) { max_size = size; }

protected:
    // total size to be accepted from child
    int max_size;

    // log itself
    string log;

    // true if child went over limit
    bool skipped;

    // write to log
    void write( string a_line, bool from_child = false );
};

class Job_Run
{
public:
    // job's status
    Exec_Status status;

    // mainly take care of pipes
    Job_Run()
        : status( Error_Status ), ex_stat( -1 )
        { in[0] = in[1] = out[0] = out[1] = rvals[0] = rvals[1] = -1; }
    ~Job_Run();

    // does all the work
    void run( Job_Order &jo );

protected:
    // values returned by child
    string r_str;

    // total size to be accepted from child
```

44

APPENDIX A
SOURCE CODE LISTING

```
int log_size;

// status as reported by waitpid (initialized to -1)
int ex_stat;

// logs used to hold output from child
Job_Log log1; // normal operation
Job_Log log2; // after SIGTERM
Job_Log log3; // after SIGKILL

// command to execute
string command;

// file descriptors of pipes to child
int in[2]; // child input pipe
int out[2]; // child output pipe (stdout & stderr)
int rvals[2]; // child return values pipe

// buffered io attached to child pipes
Pipe_Out ch_in;
Pipe_In ch_out;
Pipe_In ch_rvals;

// child's pid
pid_t child;

// total amount of time child is allowed to run
time_t timeout;

// content to be passed to child on stdin
string child_input;

// create child, do io with child, wait, send SIG's if needed
void execute_cmd();

// setup before forking, and fork
void spawn_off_child();

// read/write loop followed by wait loop
bool read_write_wait( Job_Log &s_log, time_t max_time );

// wait for at most max_time
bool wait_for_child( time_t max_time );

// read return values from child and write to r_str;
void read_rvals();

// read from child and write to log
void chld_out_to_log( Job_Log &s_log );

// code executed by child (setup and exec command)
void child_code();

// writes all it can to child
// closes pipe if nothing is left to write
void write_to_chld();

// figure out params and setup
void init( Job_Order &jo );

// assemble logs and figure out exit status
void prep_log_and_stat();

// build input to give to child
void build_child_input();

// make changes to job order record
void update_job_order( Job_Order &jo );

// parse r_str and copy to jo.r_map
void copy_rvals( Job_Order &jo );
};

XXXXXXXXXX END /share/Kiki/WF/prod/agent/Job_Run.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/agent/Pipe_IO.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/*****************************************************************
 *
 *    $Id: Pipe_IO.h,v 1.10 1998/12/14 18:48:29 rt Exp $
 *
 *    Desc: IO on pipes used to talk between agent and child
 *
 *****************************************************************/

class Pipe_Out
{
public:
    // default constructor, but call attach before using the object
    Pipe_Out()
        : fd( -1 ), out_buf( NULL ),
          out_size( 0 ), out_eof( true ),
          out_i( 0 ), closed( true ) {}

    // free buf
    ~Pipe_Out() { if ( out_buf ) delete []out_buf; }

    // write as much content as possible without blocking
    void write_some();
```

APPENDIX A
SOURCE CODE LISTING

```
    // give object content to write out
    void put_buf( string &s );

    // pipe is closed on the other end
    bool eof() { return out_eof; }

    // either pipe is closed or nothing more to write
    bool finished() { return out_eof || out_i == out_size; }

    // attach to an fd (important part of setup)
    void attach( int a_fd );

    // has it been closed
    bool is_closed() { return closed; }

    // close fd (only once)
    void close() { if ( !closed ) ::close( fd ); }
protected:
    // buffer with content
    char *out_buf;

    // beginning of unwritten content
    int out_i;

    // current size of buffer
    int out_size;

    // true if pipe closed on other end
    bool out_eof;

    // pipe's fd
    int fd;

    // true if fd is closed (our end of pipe that is)
    bool closed;
};

class Pipe_In
{
public:
    // default constructor - but call attach before using object
    Pipe_In()
        : fd( -1 ), size( 0 ), in_buf( NULL ), in_eof( true ),
          max_size( 0 ), buf_size( 0 ), beg( 0 ), end( 0 ) {}

    // free buffer
    ~Pipe_In(){ if ( in_buf ) delete []in_buf; }

    // read as much as possible without blocking end
    // within limits of max_size, return size of content
    // in buffer, and set head_ptr to poin to beginning
    // of content
    int read_some( char **head_ptr );

    // other end of pipe is closed
    bool eof() { return in_eof; }

    // move head back (but within limits) - throws x_base
    // if out of limits
    void unget( int some_num );

    // move head forwrd (but within limits) - throws x_base
    // if out of limits
    void get( int some_num );

    // buffer limit reeched (no new content will be read until
    // some content is taken from the buffer)
    bool full() { return size == max_size; }

    // attach to an fd, end set mex_size - cell before using the object
    void attach( int e_fd, int e_size );
protected:
    // true if other end of pipe is closed
    bool in_eof;

    // fd for the pipe
    int fd;

    // current size of content in buffer
    int size;

    // max size of content in buffer
    int max_size;

    // buffer size (2*max_size)
    int buf_size;

    // beginning of content in buffer
    int beg;

    // first empty space in buffer (end of content)
    int end;

    // buffer with content
    char *in_buf;

    // move content from the secont half of buffer to the first
    void move_buf();
```

APPENDIX A
SOURCE CODE LISTING

```
;;
XXXXXXXXXX END /share/Kiki/WF/prod/agent/Pipe_IO.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/engine/EngineD.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/*****************************************************************************
 *
 *      $Id: EngineD.h,v 1.13 1999/02/13 21:54:00 rt Exp $
 *
 *      Desc: Main code for engine
 *
 *****************************************************************************/


class EngineD
{
public:
  EngineD()
      : interval( 5 ), log_file( &cerr ), once( false ) {}
  ~EngineD();

  // main code for engine
  void do_work();

protected:
  LDAP_Wrap ldap;
  int interval;
  bool once;
  string server;
  string request_fltr;
  string action_fltr;
  string serviceDN;
  String_Vector action_list;
  ostream *log_file;


  void wait_for_bg_jobs( Request &req );
  bool check_on_job( Request &req, int jnum );
  void state_machine( Request &req );
  void move_out_req( Request &req );
  void place_request( Request &r, Job_Order_Vec &jov );
  void process_queued_request( LDAP_Entry &e );
  void process_new_req( Req_Builder &rb, LDAP_Entry &e );
  void place_next_order( Request &req );
  void process_action_in( int a_num );
  void process_action_queue( int a_num );
  void build_action_list();
  void build_ldap_params();
  void check_params();
  void log_setup();
  void write_to_log( string msg )
    { (*log_file) << time_stamp() << " " << msg << endl << flush; }
};
XXXXXXXXXX END /share/Kiki/WF/prod/engine/EngineD.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/engine/EngineD.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/*****************************************************************************
 *
 *      $Id: EngineD.cc,v 1.22 1999/02/17 02:30:51 rt Exp $
 *
 *      Desc: Main code for engine
 *
 *****************************************************************************/


#include "WF.h"
#include <unistd.h>
#include "EngineD.h"


// main code for agent
void EngineD::do_work()
{
  try {
    // set up logging
    log_setup();

    // get all we need to do the work
    check_params();
    build_ldap_params();

    // main work loop
    while ( true ) {
      // build action list
      build_action_list();

      // clear action incoming
      for( int i=0; i < action_list.size(); i++)
        {
          try { process_action_in( i ); }
          catch ( x_base x ) { write_to_log( x.msg ); }
        }

      // work on requests
      for( int i=0; i < action_list.size(); i++ )
        {
          try { process_action_queue( i ); }
          catch ( x_base x ) { write_to_log( x.msg ); }
        }
```

```
        // they asked us to go through queue only once
        if ( once )
            {
            write_to_log( "One pass through queue completed" );
            break;
            }

        // delay before reconnecting to ldap
        ldap.disconnect();
        sleep( interval );
        }
    }
    catch ( x_base x ) {
        write_to_log( (string)"Fatal error: " + x.msg );
        throw;
    }
    catch ( std::exception &x ) {
        write_to_log( (string)"Fatal error (stdlib): " + x.what() );
        throw;
    }
    catch ( ... ) {
        write_to_log( "Unknown fatal error." );
        throw;
    }
}

void EngineD::build_action_list()
{
    // search for actions and add them all to the queue
    LDAP_Entry_Vec rv = ldap.search(serviceDN, LDAP_SCOPE_SUBTREE, action_fltr);
    action_list.clear();
    for ( int i=0; i < rv.size(); i++ )
        action_list.push_back( rv[i].dn );
}

void EngineD::build_ldap_params()
{
    serviceDN = config.val(ServiceDN_Param);
    request_fltr = Obj_Class_Attr + "=" + Request_Obj;
    action_fltr = Obj_Class_Attr + "=" + Action_Obj;

    // set ldap prefs
    ldap.prefs( config.server, config.bindDN, config.bindPW, interval );
}

EngineD::~EngineD()
{
    write_to_log( "Exiting ..." );

    if ( log_file && log_file != &cerr )
        delete log_file;
}

void EngineD::log_setup()
{
    // are we given a log file?
    if ( config.has_a_val( Log_File_Param ) )
        {
        string log_name = config.val( Log_File_Param );

        // open log
        log_file = new std::ofstream( log_name.c_str(), ios::app );

        if ( !(*log_file) )
            throw( x_base( TPFX + "Unable to open config file " + log_name ) );
        }

    // write an opening message
    write_to_log( "Engine starting ... " );
}

void EngineD::check_params()
{
    // Require these minimum parameters
    if ( config.server.empty() )
        throw( x_base( TPFX + "No LDAP server specified" ) );

    if ( config.bindDN.empty() )
        throw( x_base( TPFX + "No bind DN specified" ) );

    if ( config.bindPW.empty() )
        throw( x_base( TPFX + "No bind password specified" ) );

    if ( !config.has_a_val(ServiceDN_Param) )
        throw( x_base( TPFX + "No service DN specified" ) );

    // OPTIONS
    // Interval to sleep between connections to LDAP
    if ( config.has_a_val( Interval_Param ) )
        interval = safe_atoi( config.val( Interval_Param ) );

    // Go through queue only once
    if ( config.peek( Once_Param ) )
        once = true;
}

void EngineD::process_action_in( int a_num )
{
```

APPENDIX A
SOURCE CODE LISTING

```
  string &actionDN = action_list[ a_num ];
  string in_dn = ID_Attr + "=" + In_Fldr + ", " + actionDN;

  // search for incoming requests
  LDAP_Entry_Vec rv(ldap.search(in_dn, LDAP_SCOPE_ONELEVEL, request_fltr));

  // no work here
  if ( rv.size() <= 0 )
    return;

  // read action info
  Req_Builder rb;
  rb.build_protos( ldap, actionDN );

  // process incoming requests
  for (int i=0; i< rv.size(); i++)
    {
      try { process_new_req( rb, rv[i] ); }
      catch ( x_base x ) { write_to_log( x.msg ); }
    }
}

void EngineD::process_new_req( Req_Builder &rb, LDAP_Entry &e )
{
  write_to_log( "Working on new request " + e.dn );
  Request Req;

  try {
    // parse incoming request
    req.init_from_entry( e );

    // build new dn for request
    Request new_req;
    new_req.id = req.id;
    new_req.parentDN = rb.req.parentDN;

    // first check if the request has been placed in the queue
    bool placed = true;
    try { new_req.init_from_ldap( ldap ); }
    catch ( x_ldap x ) {
      if ( x.err != LDAP_NO_SUCH_OBJECT )
        throw;

      placed = false;
    }

    // incomplete request has been placed before
    if ( placed && new_req.status == Hold_Status )
      {
        ldap.remove_subtree( new_req.dn );
        placed = false;
      }

    // need to place request
    if ( !placed ) {
      new_req = rb.build_req( req );
      Job_Order_Vec new_orders = rb.build_jo_vec( new_req );
      place_request( new_req, new_orders );
    }
  }
  catch ( x_base x ) {
    // some problem while parsing request
    string err = (string)"Error while processing: " + x.msg;
    req.write_to_log( err );
    req.status = Error_Status;
    write_to_log( err );

    // move out of the incoming folder
    move_out_req( req );

    return;
  }

  // request placed successfully - remove incoming request
  ldap.remove( req.dn );
}

void EngineD::place_request( Request &r, Job_Order_Vec &jov )
{
  // write request
  r.add_to_ldap( ldap );

  // write job orders
  for ( int i=0; i<jov.size(); i++ )
    jov[i].add_to_ldap( ldap );

  // change request status to runnable
  ldap.replace_attr( r.dn, Status_Attr, Runnable_Status );
}

void EngineD::process_action_queue( int a_num )
{
  string &actionDN = action_list[ a_num ];
  string q_dn = ID_Attr + "=" + Queue_Fldr + ", " + actionDN;

  // should not retrieve requests that are on hold
  string filter = Obj_Class_Attr + "=" + Request_Obj;

  // search for queued requests
  LDAP_Entry_Vec rv( ldap.search( q_dn, LDAP_SCOPE_ONELEVEL, filter) );
```

APPENDIX A
SOURCE CODE LISTING

```
    // process requests
    for (int i=0; i< rv.size() ; i++)
        {
        try { process_queued_request( rv[i] ); }
        catch ( x_base x ) { write_to_log( x.msg ); }
        }
    }

void EngineD::process_queued_request( LDAP_Entry &e )
{
    write_to_log( "Processing request " + e.dn );

    // parse request
    Request req;

    try {
        req.init_from_entry( e );
        state_machine( req );
    }
    catch ( x_base x ) {
        req.write_to_log( x.msg );
        req.status = Error_Status;
        move_out_req( req );
        return;
    }

    // make sure all changes are recorded
    if ( req.status != Complete_Status && req.status != Error_Status )
        req.update_in_ldap( ldap );
}

void EngineD::state_machine( Request &req )
{
    if ( req.status == Hold_Status )
        return;

    if ( req.status == Complete_Status || req.status == Error_Status )
        {
        move_out_req( req );
        return;
        }

    if ( req.status == Runnable_Status )
        {
        ldap.replace_attr( req.dn, Status_Attr, Running_Status );
        req.status = Running_Status;
        }

    while ( req.pc < req.states.size() )
        {
        // background job
        if ( req.states[req.pc].bg )
            {
            if ( req.states[req.pc].status == Runnable_Status )
                place_next_order( req );
            req.pc++;
            continue;
            }

        // foreground job

        // has not been placed yet
        if ( req.states[req.pc].status == Runnable_Status )
            {
            place_next_order( req );
            break;
            }

        // error occured - wait for all bg jobs
        if ( req.states[req.pc].status == Error_Status )
            break;

        // done with this one
        if ( req.states[req.pc].status == Complete_Status )
            {
            req.pc++;
            continue;
            }

        // sanity check
        if ( req.states[req.pc].status != Running_Status )
            throw ( x_base( TPFX + "Invalid status of job " + num2str(req.pc) ) );

        // running job - check on progress

        // not ready yet
        if ( !check_on_job( req, req.pc ) )
            break;
        }

    // done with fg jobs in this action - wait for bg jobs
    if ( req.pc >= req.states.size() || req.states[req.pc].status == Error_Status )
        wait_for_bg_jobs( req );

    // all bg jobs are done
    if ( req.status == Complete_Status || req.status == Error_Status )
        {
        move_out_req( req );
        return;
```

APPENDIX A
SOURCE CODE LISTING

```
  \
}

void EngineD::move_out_req( Request &req )
{
  req.update_in_ldap( ldap );

  // build new dn
  string olddn = req.dn;
  req.parentDN = ID_Attr + '=' + Out_Fldr + ", " + req.actionDN;
  req.dn = ID_Attr + '=' + req.id + ", " + req.parentDN;
  ldap.move_subtree( olddn, req.dn );
}

void EngineD::wait_for_bg_jobs( Request &req )
{
  int i;

  for (i=0; i<req.pc; i++)
    {
      // for each background running job
      if ( req.states[i].bg && req.states[i].status == Running_Status )
        // not ready yet
        if ( !check_on_job( req, i ) )
          break;
    }

  // some bg job not ready yet
  if ( i<req.pc )
    return;

  // all bg jobs are done - figure out request status
  req.status = Complete_Status;
  for (i=0; i<req.states.size(); i++)
    // this job has failed - so request failed
    if ( req.states[i].status != Complete_Status )
      {
        req.status = Error_Status;
        string msg = (string)"Job number " + num2str(i) + " failed.";
        req.write_to_log(msg);
        break;
      }
}

bool EngineD::check_on_job( Request &req, int a_pc )
{
  Job_Order jo_engn;
  Job_Order jo_agnt;

  // read engine and agent's copies of job order
  jo_engn.id = req.id + 'j' + num2str( a_pc );
  jo_engn.parentDN = req.dn;
  jo_engn.init_from_ldap( ldap );
  jo_agnt.id = jo_engn.id;
  jo_agnt.parentDN = jo_engn.agent_dn;

  // see if agent's copy exists
  bool ex = true;
  try { jo_agnt.init_from_ldap( ldap ); }
  catch (x_ldap x) { if (x.err != LDAP_NO_SUCH_OBJECT) throw; ex=false; }

  // agent's copy exists
  if ( ex )
    {
      // not ready yet
      if (jo_agnt.status != Error_Status && jo_agnt.status != Complete_Status)
        return false;

      // have not pulled changes yet
      if ( jo_engn.status != jo_agnt.status )
        {
          jo_engn.status = jo_agnt.status;
          jo_engn.log = jo_agnt.log;
          jo_engn.r_map = jo_agnt.r_map;
          jo_engn.update_in_ldap( ldap );

          // if completed successfully pull return values
          if ( jo_engn.status == Complete_Status )
            Req_Builder::pull_ret_vals(req, jo_engn, a_pc);
        }

      // remove from agent's queue
      ldap.remove( jo_agnt.dn );
    }

  // update job status in request
  req.states[a_pc].status = jo_engn.status;

  // job is complete (or errored out)
  return true;
}

void EngineD::place_next_order( Request &req )
{
  // get engine's copy of job order
  Job_Order jo;
  jo.id = req.id + 'j' + num2str( req.pc );
  jo.parentDN = req.dn;
  jo.init_from_ldap( ldap );
```

APPENDIX A
SOURCE CODE LISTING

```
// get agent's copy of job order
Job_Order jo_agnt;
jo_agnt.id = jo.id;
jo_agnt.parentDN = jo.agent_dn;
bool ex = true;
try { jo_agnt.init_from_ldap( ldap );}
catch { x_ldap x } { if ( x.err != LDAP_NO_SUCH_OBJECT ) throw; ex = false; }

// job order has not been placed before
if ( !ex )
   {
     // figure out param values
     Req_Builder::prep_jo(req, jo, req.pc);
     jo.update_in_ldap( ldap );

     // now place it in agent's queue
     jo.status = Runnable_Status;
     jo.parentDN = jo.agent_dn;
     jo.dn = "";
     jo.add_to_ldap( ldap );
   }

// change job order status in request
req.states[req.pc].status = Running_Status;
}

XXXXXXXXXX END /share/Kiki/WF/prod/engine/EngineD.cc XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/engine/engined.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/*******************************************************************
 *
 *       $Id: engined.cc,v 1.5 1999/02/03 09:49:34 rt Exp $
 *
 *       Desc: engine - the least of it
 *
 *******************************************************************/

#include "WF_ext.h"
#include "EngineD.h"

main( int argc, char **argv )
{
  try {
     config.init( argc, argv );
     EngineD engine;
     engine.do_work();
  }
  catch( x_base x ) { die( x.msg ); }
  catch( std::exception &x ) { die( x.what() ); }
  catch( ... ) { die("Unknown fatal error."); }

  exit( 0 );
}
XXXXXXXXXX END /share/Kiki/WF/prod/engine/engined.cc XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/ui/get_obj.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/*******************************************************************
 *
 *       $Id: get_obj.cc,v 1.8 1999/02/14 02:52:27 rt Exp $
 *
 *       Desc: get an object
 *
 *******************************************************************/

#include "WF_ext.h"

LDAP_Wrap ldap;

void check_params();
void read_ldap();

int main (int argc, char **argv)
{
  try {
     config.init( argc, argv );
     check_params();
     read_ldap();
  }
  catch( x_base x ) { die( x.msg ); }
  catch( std::exception &x ) { die( x.what() ); }
  catch( ... ) { die("Unknown fatal error." ); }

  exit( 0 );
}
void read_ldap()
{
  string objDN = config.val( ObjDN_Param );
  string filter = config.val( Filter_Param );
  int scope = safe_atoi( config.val( Scope_Param ) );

  // search for objects
  LDAP_Entry_Vec rv = ldap.search( objDN, scope, filter );

  // output results
  Obj *o = NULL;
  for ( int i=0; i<rv.size(); i++ )
    {
      try {
```

APPENDIX A
SOURCE CODE LISTING

```
        o = Obj::make_obj_from_entry( rv[i] );
        cout << (*o).print_url() << endl;
      }
      catch( x_base x ) {
        Broken b;
        b.init_from_entry( rv[i], x.msg );
        cout << b.print_url() << endl;
      }
    }
}
void check_params()
{
  // need objDN
  if ( !config.has_a_val( ObjDN_Param ) )
    throw( x_base( TPFX + "No object DN specified" ) );

  // need LDAP server
  if ( config.server.empty() )
    throw( x_base( TPFX + "No LDAP server specified" ) );

  // search filter - optional
  if ( !config.has_a_val( Filter_Param ) )
    config.add_val( Filter_Param, Default_Filter );

  // search scope - optional
  if ( !config.has_a_val( Scope_Param ) )
    config.add_val( Scope_Param, LDAP_SCOPE_BASE );

  // set up ldap preferences
  ldap.prefs( config.server, config.bindDN, config.bindPW, -1);
}
XXXXXXXXXX END /share/Kiki/WF/prod/ui/get_obj.cc XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/ui/move_obj.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/
/**********************************************************************
 *
 *    $Id: move_obj.cc,v 1.4 1999/02/14 02:55:09 rt Exp $
 *
 *    Desc: copy/move/delete an object (and subtree beneath it)
 *
 **********************************************************************/

#include "WF_ext.h"

LDAP_Wrap ldap;
Cmd_Enum cmd;
void check_params();
void read_write();

int main (int argc, char **argv)
{
  try {
    config.init( argc, argv );
    check_params();
    read_write();
  }
  catch( x_base x ) { die( x.msg ); }
  catch( std::exception &x ) { die( x.what() ); }
  catch( ... ) { die( "Unknown fatal error." ); }

  exit( 0 );
}
void read_write()
{
  string &objDN = config.val( ObjDN_Param );

  // if command is delete - blast the whole subtree
  if ( cmd == Del_Cmd ) {
    ldap.remove_subtree( objDN );
    return;
  }

  // moving is also simple
  if ( cmd == Move_Cmd ) {
    ldap.move_subtree( objDN, config.val( TargetDN_Param ) );
    return;
  }

  // copying - not implemented yet
  if ( cmd == Copy_Cmd ) {
    throw( x_base( TPFX + "Copy not implemented yet" ) );
  }

  // should not get here
  throw( x_base( TPFX + "Unknown command" ) );
}
void check_params()
{
  // need bindDN and bindPW
  if ( config.bindDN.empty() )
    throw( x_base( TPFX + "No bind DN specified" ) );
  if ( config.bindPW.empty() )
    throw( x_base( TPFX + "No bind PW specified" ) );

  // need objDN
  if ( !config.has_a_val( ObjDN_Param ) )
    throw( x_base( TPFX + "No object DN specified" ) );

  // need command to perform
```

APPENDIX A
SOURCE CODE LISTING

```
if ( !config.has_a_val( Cmd_Param ) )
  throw( x_base( TPFX + "No command specified" ) );
string &c = config.val(Cmd_Param);
int i;
for(i=0; i<Cmd_Enum_Size; i++) {
  if ( c == Cmd_Enum_Name[i] ) {
    cmd = (Cmd_Enum)i;
    break;
  }
}
if ( i >= Cmd_Enum_Size )
  throw( x_base( TPFX + "Unknown command: " + c ) );

// need target DN if cmd is copy or move
if ( cmd == Move_Cmd || cmd == Copy_Cmd ) {
  if ( !config.has_a_val( TargetDN_Param ) )
    throw( x_base( TPFX + "Need target DN for command " + c ) );
}

// need LDAP server
if ( config.server.empty() )
  throw( x_base( TPFX + "No LDAP server specified" ) );

// // scope - optional
// if ( !config.has_a_val( Scope_Param ) )
//   config.add_val( Scope_Param, LDAP_SCOPE_BASE );

// set up ldap preferences
ldap.prefs( config.server, config.bindDN, config.bindPW, -1);
}
XXXXXXXXXX END /share/Kiki/WF/prod/ui/move_obj.cc XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/ui/run_action.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/
/*****************************************************************
 *
 *    $Id: run_action.cc,v 1.27 1999/01/22 09:06:29 rt Exp $
 *
 *    Desc: submit a request to run an action
 *
 *****************************************************************/

#include "WF_ext.h"

void check_params();

Request req;
LDAP_Wrap ldap;

int main (int argc, char **argv)
{
  try
  {
    // Get parameters
    config.init(argc, argv);

    // check we got all we need
    check_params();

    // make request
    req.generate_new_req(config.val(ObjDN_Param), config.user_map, ldap);

    // Send in the request
    req.add_to_ldap( ldap );

    // print out request
    cout << req.print_url() << endl;
  }
  catch ( x_base x ) { die( x.msg ); }
  catch ( std::exception &x ) {
    die( (string)"Fatal error (stdlib): " + x.what() );
    throw;
  }
  catch ( ... ) {
    die( "Unknown fatal error." );
    throw;
  }

  exit( 0 );
}


void check_params()
{
  // Figure out actionDN, server, and bind params
  if ( !config.has_a_val( ObjDN_Param ) )
    throw( x_base( TPFX + "No actionDN specified" ) );

  if ( config.server.empty() )
    throw( x_base( TPFX + "No LDAP server specified" ) );

  if ( config.bindDN.empty() )
    throw( x_base( TPFX + "No bind DN specified" ) );

  if ( config.bindPW.empty() )
    throw( x_base( TPFX + "No bind password specified" ) );

  ldap.prefs( config.server, config.bindDN, config.bindPW, -1 );
}
XXXXXXXXXX END /share/Kiki/WF/prod/ui/run_action.cc XXXXXXXXXX
```

APPENDIX A
SOURCE CODE LISTING

```
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/ui/update_obj.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/*.............................................................................
 *
 *      $Id: update_obj.cc,v 1.3 1999/02/01 07:05:55 rt Exp $
 *
 *    Desc: replace objects attrs in LDAP (the object has to exist)
 *
 ..............................................................................*/

#include "WF_ext.h"

LDAP_Wrap ldap;
bool newobj = false;

void check_params();
void read_write();

int main (int argc, char **argv)
{
  try {
    config.init( argc, argv );
    check_params();
    read_write();
  }
  catch( x_base x ) { die( x.msg ); }
  catch( std::exception &x ) { die( x.what() ); }
  catch( ... ) { die( "Unknown fatal error." ); }

  exit( 0 );
}
void read_write()
{
  // Parse user input
  LDAP_Entry e;
  e.init_from_url( config.val(Obj_Param) );

  // Create new object
  if ( newobj ) {
    string type = "type";
    if ( !e.has_a_val(type) )
      throw(x_base(TPFX+"No type specified for new object"));
    Obj *o = Obj::make_obj(e.val(type));
    o->init_from_entry(e);
    o->add_to_ldap(ldap);
    return;
  }

  // Update existing object

  // search for the object
  LDAP_Entry_Vec rv = ldap.search( e.dn, LDAP_SCOPE_BASE );
  if ( rv.size() != 1 )
    throw(x_base(TPFX+"Invalid number of search results"));
  LDAP_Entry &e2 = rv[0];

  // Parse the retrieved object
  Obj *o = NULL;
  o = Obj::make_obj_from_entry( e2 );

  // now try to make the new version of the object
  for ( LDAP_Entry::iterator i=e.begin(); i!=e.end(); i++ )
    e2[i->first] = i->second;
  Obj *n = NULL;
  n = Obj::make_obj_from_entry( e2 );

  // Finally write to LDAP
  n->update_in_ldap(ldap);
}
void check_params()
{
  // need obj
  if ( !config.has_a_val( Obj_Param ) )
    throw( x_base( TPFX + "No object specified" ) );

  // need LDAP server
  if ( config.server.empty() )
    throw( x_base( TPFX + "No LDAP server specified" ) );

  // bind parameters - required
  if ( config.bindDN.empty() )
    throw( x_base( TPFX + "No bind DN specified" ) );
  if ( config.bindPW.empty() )
    throw( x_base( TPFX + "No bind password specified" ) );

  // new object? (optional)
  if ( config.peek( New_Param ) )
    newobj = true;

  // set up ldap preferences
  ldap.prefs( config.server, config.bindDN, config.bindPW, -1);
}
XXXXXXXXXX END /share/Kiki/WF/prod/ui/update_obj.cc XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Action.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/*.............................................................................
 *
 *      $Id: Action.cc,v 1.24 1999/02/17 05:18:25 rt Exp $
 *
```

APPENDIX A
SOURCE CODE LISTING

```
 *
 *     Desc: Action class
 ***********************************************************************/

#include "WF.h"

void Action::init_from_entry( LDAP_Entry &e )
{
  Obj::init_from_entry( e );

  // formURL
  if ( e.has_a_val( URL_Attr ) )
    formURL = e.val( URL_Attr );

  // script
  if ( !e.has_a_val( Script_Attr ) )
    throw ( x_base( TPFX + "No script attr" ) );
  script.init( e.val(Script_Attr) );

  // param
  if ( e.has_a_val( Param_Attr ) )
    p_vec = e[Param_Attr];
}

string Action::print_url()
{
  string ret = Obj::print_url();

  // param list
  for ( int i=0; i<p_vec.size(); i++ )
    ret += (string)"&param=" + url_encode( p_vec[i] );

  // script
  ret += (string)"&script=" + script.print_url();

  // formURL
  ret += (string)"&formurl=" + url_encode( formURL );

  return ret;
}

LDAP_Entry Action::make_entry()
{
  LDAP_Entry e = Obj::make_entry();
  e[ Param_Attr ] = p_vec;
  e[ URL_Attr ] = make_vector( formURL );
  e[ Script_Attr ] = make_vector( script.print_str() );
  return e;
}

void Action::add_to_ldap( LDAP_Wrap &ldap )
{
  // first add the action record itself
  Obj::add_to_ldap( ldap );

  // now create folders for queues
  Folder in, queue, out;

  // for incoming requests
  in.id = In_Fldr;
  in.parentDN = dn;
  in.add_to_ldap( ldap );

  // for requests being processed
  queue.id = Queue_Fldr;
  queue.parentDN = dn;
  queue.add_to_ldap( ldap );

  // for completed requests
  out.id = Out_Fldr;
  out.parentDN = dn;
  out.add_to_ldap( ldap );
}

XXXXXXXXXX END /share/Kiki/WF/prod/util/Action.cc XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Exception.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/**********************************************************************
 *
 *     $Id: Exception.cc,v 1.6 1999/02/14 02:46:24 rt Exp $
 *
 *     Desc: exception handling
 *
 **********************************************************************/

#include "WF.h"
#include <errno.h>
#include <string.h>
#include <netdb.h>

char *HER[]={"SUCCESS","HOST_NOT_FOUND","TRY_AGAIN","NO_RECOVERY","NO_DATA"};

x_net::x_net( string u_msg ) {
  if ( h_errno == -1 )
    x_sys( u_msg );
  else if ( h_errno > 4 )
    x_base( u_msg + "Unknown network error.", h_errno );
  else
```

```
    x_base( u_msg + HER(h_errno), h_errno );
}

x_sys::x_sys( string u_msg )
{
  char *tmp = strerror( errno );

  if ( tmp )
    x_base( u_msg + tmp, errno );
  else
    x_base( u_msg + "Unknown system error.", errno );
}
XXXXXXXXXX END /share/Kiki/WF/prod/util/Exception.cc XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Job_Order.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/*****************************************************************************
 *
 *     $Id: Job_Order.cc,v 1.12 1999/02/21 04:28:56 rt Exp $
 *
 *     Desc: Job_Order implementation
 *
 *****************************************************************************/

#include "WF.h"

LDAP_Entry Job_Order::make_entry()
{
  LDAP_Entry e = Obj::make_entry();

  // status
  e[ Status_Attr ] = make_vector( status );

  // agentDN
  e[ AgentDN_Attr ] = make_vector( agent_dn );

  // jobDN
  e[ JobDN_Attr ] = make_vector( job_dn );

  // actionDN
  e[ ActionDN_Attr ] = make_vector( action_dn );

  // param
  if ( p_map.size() > 0 )
    e[ Param_Attr ] = p_map.make_assign_vector();
  else
    e[ Param_Attr ] = make_empty_vector();

  // rvals
  if ( r_map.size() > 0 )
    e[ Rval_Attr ] = r_map.make_assign_vector();
  else
    e[ Rval_Attr ] = make_empty_vector();

  // log
  e[ Log_Attr ] = make_vector( log );

  // command
  e[ Command_Attr ] = make_vector( command );

  return e;
}

void Job_Order::init_from_entry( LDAP_Entry &e )
{
  Obj::init_from_entry( e );

  // status
  if ( !e.has_a_val( Status_Attr ) )
    throw ( x_base( TPFX + "No status attr in job order " + dn ) );
  status = get_exec_status( e.val( Status_Attr ) );

  // log
  if ( e.has_a_val( Log_Attr ) )
    log = e.val( Log_Attr );

  // command
  if ( e.has_a_val( Command_Attr ) )
    command = e.val( Command_Attr );

  // agentDN
  if ( !e.has_a_val( AgentDN_Attr ) )
    throw( x_base( TPFX + "No agentDN attr in job order " + dn ) );
  agent_dn = e.val( AgentDN_Attr );

  // actionDN
  if ( !e.has_a_val( ActionDN_Attr ) )
    throw( x_base( TPFX + "No action DN attr in job order " + dn ) );
  action_dn = e.val( ActionDN_Attr );

  // jobDN
  if ( !e.has_a_val( JobDN_Attr ) )
    throw( x_base( TPFX + "No Job DN attr in job order " + dn ) );
  job_dn = e.val( JobDN_Attr );

  // params
  if ( e.has_a_val( Param_Attr ) )
    p_map.parse_assign_vector( e[ Param_Attr ] );

  // rvals
```

APPENDIX A
SOURCE CODE LISTING

```
  if ( e.has_a_val( Rval_Attr ) )
    r_map.parse_assign_vector( e[ Rval_Attr ] );
}

string Job_Order::print_url()
{
  string ret = Obj::print_url();

  // status
  ret += "&status=";
  ret += Exec_Status_Name( status );

  // param
  String_Vector tmp = p_map.make_assign_vector();
  for ( int i=0; i<tmp.size(); i++ )
    ret += (string)"&param=" + url_encode( tmp[i] );

  // rvals
  tmp = r_map.make_assign_vector();
  for ( int i=0; i<tmp.size(); i++ )
    ret += (string)"&rval=" + url_encode( tmp[i] );

  // log
  if ( !log.empty() )
    ret += (string)"&log=" + url_encode( log );

  // command
  ret += (string)"&command=" + url_encode( command );

  // agentDN
  ret += (string)"&agentdn=" + url_encode( agent_dn );

  // actionDN
  ret += (string)"&actiondn=" + url_encode( action_dn );

  // jobDN
  ret += (string)"&jobdn=" + url_encode( job_dn );

  return ret;
}
XXXXXXXXXX END /share/Kiki/WF/prod/util/Job_Order.cc XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Job_State.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/*******************************************************************************
 *
 *    $Id: Job_State.cc,v 1.1 1998/12/16 00:42:16 rt Exp $
 *
 *    Desc: Job_State implementation
 *
 *******************************************************************************/

#include "WF.h"

string Job_State::print_url()
{
  string ret;
  if ( bg )
    ret += "BG";
  else
    ret += "FG";

  ret += "+";
  ret += Exec_Status_Name( status );
  return ret;
}
XXXXXXXXXX END /share/Kiki/WF/prod/util/Job_State.cc XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/LDAP_Entry.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/*******************************************************************************
 *
 *    $Id: LDAP_Entry.cc,v 1.18 1999/01/18 20:54:56 rt Exp $
 *
 *    Desc: Map of attrs to values - used to pass info about LDAP records
 *
 *******************************************************************************/

#include "WF.h"


void LDAP_Entry::init( LDAP *ld, LDAPMessage *e )
{
  BerElement *ber;
  char *dn_str;
  char *a;

  // Get entry's dn
  dn_str = ldap_get_dn( ld, e );
  dn = dn_str;
  ldap_memfree( dn_str );

  // For each attribute of the entry
  for ( a = ldap_first_attribute( ld, e, &ber );
        a != NULL;
        a = ldap_next_attribute( ld, e, ber ) )
  {
    // Turn attribute name lowercase
    string tmp( downcase( a ) );
```

APPENDIX A
SOURCE CODE LISTING

```
    // Associate values vector with attribute name
    operator[]( tmp ) = parse_values( ld, e, a );
    ldap_memfree( a );
  }

  // Cleanup
  if ( ber != NULL )
    {
      ber_free( ber, 0 );
    }
}

// Construct a vector of values
String_Vector LDAP_Entry::parse_values ( LDAP *ld, LDAPMessage *e, char *e )
{
  String_Vector ret;
  char **vals;
  int i;

  if ((vals = ldap_get_values( ld, e, a)) == NULL )
    return ret;

  for ( i = 0; vals[i] != NULL; i++ )
    ret.push_back(vals[i]);

  ldap_value_free( vals );

  return ret;
}

// prints in LDIF format
string LDAP_Entry::print_str()
{
  string ret;
  int i, num_vals;

  ret += "dn: " + dn + "\n";

  // TODO: we need to be careful to add space in front of new lines
  // if an attribute value has multiple lines
  for (iterator iter = begin(); iter != end(); iter++ )
    {
      num_vals = (*iter).second.size();
      for (i=0; i < num_vals; i++)
          ret += (*iter).first + ": " + (*iter).second[i] + "\n";
    }

  // Signal end of entry by empty line
  ret += "\n";

  return ret;
}

void LDAP_Entry::init_from_url( const string &url )
{
  parse_url_enc_params( url );
  if ( !has_a_val( DN_Attr ) )
    throw( x_base(TPFX+"No entry dn specified") );
  dn = val( DN_Attr );
}
XXXXXXXXXX END /share/Kiki/WF/prod/util/LDAP_Entry.cc XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/LDAP_Wrap.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/***********************************************************************
 *
 *      $Id: LDAP_Wrap.cc,v 1.48 1999/02/03 15:56:19 rt Exp $
 *
 *      Desc: LDAP_Wrap talks to LDAP API library
 *
 ***********************************************************************/


#include "WF.h"

// for sleep
#include <unistd.h>

void LDAP_Wrap::prefs( const string &a_host, const string &bindDN,
                       const string &bindPW, int an_interval )
{
  host = a_host;
  bind_dn = bindDN;
  bind_pw = bindPW;
  interval = an_interval;

  // if we are connected - disconnect so new settings could take effect
  if ( ld )
    disconnect();
}

void LDAP_Wrap::disconnect()
{
  int r;

  if ( !ld )
    return;

  r = ldap_unbind_s ( ld );
  ld = NULL;
```

APPENDIX A
SOURCE CODE LISTING

```
    // ignore lost connections if iterval is set
    if ( interval >=0 && ( r == LDAP_SERVER_DOWN || r == LDAP_CONNECT_ERROR ) )
        return;

    // bad stuff happened
    if ( r != LDAP_SUCCESS )
        throw( x_ldap( r, TPFX + "ldap_unbind_s: ") );
}

void LDAP_Wrap::bind()
{
    int r;

    while ( true )
    {
        // first free ld structure if it exists
        disconnect();

        // now make new one
        ld = ldap_init( host.c_str(), LDAP_PORT );
        if ( !ld )
            throw( x_sys( TPFX + "ldap_init: " ) );

        // attempt to bind
        r = ldap_simple_bind_s( ld, bind_dn.c_str(), bind_pw.c_str() );

        // no reconnect is set
        if ( interval < 0 )
            break;

        // something other than unreachable server
        if ( r != LDAP_SERVER_DOWN && r != LDAP_CONNECT_ERROR )
            break;

        // sleep before trying again
        sleep( interval );
    }

    if ( r != LDAP_SUCCESS )
        throw( x_ldap( r, TPFX + "While binding to LDAP as [" + bind_dn + "]: ") );
}

LDAP_Entry_Vec LDAP_Wrap::search( const string &base, int scope,
                                   const string &filter )
{
    LDAPMessage *res;
    int r;

    // first time
    if ( !ld )
        bind();

    while (true)
    {
        // Do the search
        r = ldap_search_ext_s( ld, base.c_str(), scope, filter.c_str(), NULL, 0,
                               NULL, NULL, NULL, 0, &res);

        // no more searching
        if ( interval<0 )
            ( r != LDAP_SERVER_DOWN && r != LDAP_CONNECT_ERROR ) )
            break;

        // lost connection - try again later
        sleep( interval );
        bind();
    }

    if ( r != LDAP_SUCCESS )
        throw( x_ldap( r, TPFX + "While searching for [" +
                       base + "] with filter [" + filter + "]: " ) );

    // Parse results
    LDAP_Entry_Vec rv = parse_res_chain( res );
    ldap_msgfree( res );

    return rv;
}

LDAP_Entry_Vec LDAP_Wrap::parse_res_chain( LDAPMessage *res )
{
    LDAP_Entry_Vec ret;
    LDAPMessage *e;

    // For each entry in result chain
    for ( e = ldap_first_entry( ld, res );
          e != NULL;
          e = ldap_next_entry( ld, e ) )
    {
        LDAP_Entry ent( ld, e );
        ret.push_back( ent );
    }

    return ret;
}

void LDAP_Wrap::remove( const string &a_dn )
{
```

APPENDIX A
SOURCE CODE LISTING

```
int r;

// first time
if ( !ld )
   bind();

while (true)
   {
     // delete the entry
     r = ldap_delete_ext_s( ld, a_dn.c_str(), NULL, NULL );

     // no more tries
     if ( interval<0 ||
            ( r != LDAP_SERVER_DOWN && r != LDAP_CONNECT_ERROR ) )
        break;

     // lost connection - try again later
     sleep( interval );
     bind();
   }

  if ( r != LDAP_SUCCESS )
     throw( x_ldap( r, TPFX + "While deleting entry [" + a_dn + "]: " ) );
}

void LDAP_Wrap::remove_subtree( const string &a_dn )
{
  LDAP_Entry_Vec rv = search( a_dn, LDAP_SCOPE_SUBTREE );
  for( int i=rv.size()-1; i>=0; i-- )
     remove( rv[i].dn );
}

void LDAP_Wrap::add( const string &a_dn, LDAPMod_NTA &mods )
{
  int r;

  // first time
  if ( !ld )
     bind();

  while (true)
     {
       r = ldap_add_ext_s( ld, a_dn.c_str(), mods, NULL, NULL );

       // no more tries
       if ( interval<0 ||
              ( r != LDAP_SERVER_DOWN && r != LDAP_CONNECT_ERROR ) )
          break;

       // lost connection - try again later
       sleep( interval );
       bind();
     }

  if ( r != LDAP_SUCCESS )
     throw( x_ldap( r, TPFX + "While adding entry [" + a_dn + "]: " ) );
}

void LDAP_Wrap::add_entry( LDAP_Entry &e )
{
  LDAPMod_NTA mods;
  LDAP_Entry::iterator i;
  int j;
  for ( i=e.begin(); i!=e.end(); i++ )
     {
       // skip certain attributes
       if ( i->first == "creatorsname" ||
               i->first == "modifiersname" ||
               i->first == "createtimestamp" ||
               i->first == "modifytimestamp" )
          continue;

       // create NTA of attr values
       Char_Star_NTA vals( i->second );

       // add to mods
       mods.push_back( LDAP_MOD_ADD, i->first.c_str(), vals );
     }

  // add to LDAP
  add( e.dn, mods );
}

void LDAP_Wrap::modify_attr( const string &a_dn, const string &attr,
                             const string &val, int mod_type )
{
  LDAPMod_NTA mods;
  Char_Star_NTA tmp;
  tmp.push_back( val );
  mods.push_back( mod_type, attr.c_str(), tmp );
  modify( a_dn, mods );
}

void LDAP_Wrap::modify( const string &a_dn, LDAPMod_NTA &mods )
{
  int r;

  // first time
  if ( !ld )
     bind();
```

61

APPENDIX A
SOURCE CODE LISTING

```
  while (true)
  {
      r = ldap_modify_ext_s( ld, a_dn.c_str(), mods, NULL, NULL );

      // no more tries
      if ( interval<0 ||
              ( r != LDAP_SERVER_DOWN && r != LDAP_CONNECT_ERROR ) )
          break;

      // lost connection - try again later
      sleep( interval );
      bind();
  }

  if ( r != LDAP_SUCCESS )
    throw( x_ldap( r, TPFX + "While modifying entry [" + a_dn + "]: ") );
}

bool LDAP_Wrap::exists( const string &a_dn )
{
  try {
    LDAP_Entry_Vec rv = search( a_dn, LDAP_SCOPE_BASE );
  }
  catch ( x_ldap x ) {
    if ( x.err != LDAP_NO_SUCH_OBJECT )
      throw;
    return false;
  }
  return true;
}

void LDAP_Wrap::move_subtree( const string &from, const string &to )
{
  // remove subtree at destination if it exists
  if ( exists( to ) )
    remove_subtree( to );

  // copy subtree
  LDAP_Entry_Vec rv = search( from, LDAP_SCOPE_SUBTREE );
  for ( int i=0; i<rv.size(); i++ )
  {
      string::size_type bag = rv[i].dn.rfind( from );
      rv[i].dn.replace( bag, from.size(), to );
      add_entry( rv[i] );
  }

  // remove old copy of subtree
  remove_subtree( from );
}

void LDAP_Wrap::update_entry( LDAP_Entry &a )
{
  LDAPMod_NTA mods;
  LDAP_Entry::iterator i;
  int j;
  for ( i=a.begin(); i!=a.end(); i++ )
  {
      // skip certain attributes
      if ( i->first == "creatorsname" ||
              i->first == "modifiersname" ||
              i->first == "createtimestamp" ||
              i->first == "modifytimestamp" ||
              i->first == "dn" )
          continue;

      // create NTA of attr values
      Char_Star_NTA vals( i->second );

      // add to mods
      mods.push_back( LDAP_MOD_REPLACE, i->first.c_str(), vals );
  }

  // update in LDAP
  modify( a.dn, mods );
}
XXXXXXXXX END /share/Kiki/WF/prod/util/LDAP_Wrap.cc XXXXXXXXX
XXXXXXXXX BEGIN /share/Kiki/WF/prod/util/LDAP_related.cc XXXXXXXXX
/*-*- Mode: C++; -*-*/

/****************************************************************
 *
 *     $Id: LDAP_related.cc,v 1.19 1998/12/14 19:12:14 rt Exp $
 *
 *     Desc: Aid in handling of LDAPMod and NULL term char* arrays -
 *           to be used by LDAP_Wrap only
 *
 ****************************************************************/

#include "WF.h"

Char_Star_NTA::Char_Star_NTA( int in_size )
  :size( in_size ), array( new (char *)[ in_size ] ), last( 0 )
{
  array[0] = NULL;
}

Char_Star_NTA::Char_Star_NTA( String_Vector &v )
  :size( v.size()+1 ), array( new (char *)[v.size()+1] ), last( v.size() )
```

APPENDIX A
SOURCE CODE LISTING

```
{
  for (int i=0;i<last;i++) {
    array[i]=strdup(v[i].c_str());
    if ( !array[i] )
      throw( x_base( TFFX + "Insufficant memory." ) );
  }
  array[last] = NULL;
}

Char_Star_NTA::Char_Star_NTA( const Char_Star_NTA &c )
  :size(c.size), array( new (char *)[ c.size ] ), last(c.last)
{
  for (int i=0;i<last;i++) {
    array[i]=strdup(c.array[i]);
    if ( !array[i] )
      throw( x_base( TFFX + "Insufficant memory." ) );
  }
  array[last] = NULL;
}

Char_Star_NTA::~Char_Star_NTA()
{
  for ( int i=0; array[i]; i++ )
    delete [](array[i]);

  delete []array;
}

void Char_Star_NTA::push_back( const char *elt )
{
  if ( !elt )
    return;

  char *elt_copy = dup_c_str( elt );

  if ( last == (size-1) )
    extend();

  array[ last++ ] = elt_copy;
  array[ last ] = NULL;
}

void Char_Star_NTA::extend()
{
  int i;
  char **old_array = array;
  size += 10;
  array = new (char *)[ size ];

  for ( i=0; old_array[i]; i++ )
    array[i] = old_array[i];
  array[i] = NULL;

  delete []old_array;
}

Char_Star_NTA::operator char **()
{
  return array;
}

char **Char_Star_NTA::dup_css( char **c )
{
  int i, n;
  for (n=0; c[n]; n++);

  char **ret = new (char *) [n+2];
  for (i=0; i<n; i++)
    ret[i] = dup_c_str( c[i] );

  ret[i] = NULL;

  return ret;
}
void Char_Star_NTA::destroy_css( char **c )
{
  int i;

  for (i=0; c[i]; i++)
    delete [](c[i]);

  delete []c;
}

//*********************************
// LDAPMod_NTA implemantation below
//*********************************

LDAPMod_NTA::LDAPMod_NTA( int in_size )
  :size( in_size ), array( new (LDAPMod *)[ in_size ] ), last( 0 )
{
  array[0] = NULL;
}

LDAPMod_NTA::LDAPMod_NTA( const LDAPMod_NTA &c )
  :size(c.size), array( new (LDAPMod *)[ c.size ] ), last(c.last)
{
  for ( int i=0; i<last; i++ )
    array[i] = LDAPMod_dup( c.array[i] );
```

APPENDIX A
SOURCE CODE LISTING

```
  array[last] = NULL;
)

LDAPMod_NTA::~LDAPMod_NTA()
{
  for ( int i=0; array[i]; i++ )
    LDAPMod_destroy( array[i] );

  delete []array;
)

void LDAPMod_NTA::push_back( int op, const char *type, char **values )
{
  if ( last == size-1 )
    extend();

  char *type_copy = dup_c_str( type );

  LDAPMod *tmp = new LDAPMod;
  tmp->mod_op = op;
  tmp->mod_type = type_copy;
  tmp->mod_values = Char_Star_NTA::dup_css( values );

  array[ last++ ] = tmp;
  array[ last ] = NULL;
)

LDAPMod_NTA::operator LDAPMod **()
{
  return array;
)

void LDAPMod_NTA::extend()
{
  int i;

  LDAPMod **old_array = array;
  size += 10;
  array = new (LDAPMod *)[ size ];

  for ( i=0; old_array[i]; i++ )
    array[i] = old_array[i];

  array[i] = NULL;

  delete []old_array;
)

void LDAPMod_NTA::LDAPMod_destroy( LDAPMod *m )
{
  delete []( m->mod_type );
  Char_Star_NTA::destroy_css( m->mod_values );
  delete m;
)
LDAPMod *LDAPMod_NTA::LDAPMod_dup( const LDAPMod *m )
{
  if ( !m )
    return NULL;

  LDAPMod *ret = new LDAPMod;
  ret->mod_op = m->mod_op;
  ret->mod_type = dup_c_str( m->mod_type );
  ret->mod_values = Char_Star_NTA::dup_css( m->mod_values );

  return ret;
)
XXXXXXXXXX END /share/Kiki/WF/prod/util/LDAP_related.cc XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Obj.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/**************************************************************************
 *
 *     $Id: Obj.cc,v 1.19 1999/02/17 05:20:44 rt Exp $
 *
 *     Desc: Class Obj is parent to all objects
 *
 **************************************************************************/

#include "WF.h"

void Obj::init_from_ldap( LDAP_Wrap &ldap )
{
  // need to build dn first
  if ( dn.empty() )
    {
      if ( id.empty() || parentDN.empty() )
           throw( x_base( TPFX + "Can not figure out dn" ) );
      dn = ID_Attr + '=' + id + ", " + parentDN;
    }

  // build filter
  string fltr = Obj_Class_Attr + '=' + type;

  // search for the object
  LDAP_Entry_Vec rv = ldap.search( dn, LDAP_SCOPE_BASE, fltr );

  // parse found entry
  init_from_entry( rv[0] );
)
```

APPENDIX A
SOURCE CODE LISTING

```cpp
void Obj::init_from_url( string a_url )
{
  LDAP_Entry e;
  e.init_from_url( a_url );
  init_from_entry( e );
}

void Obj::add_to_ldap( LDAP_Wrap &ldap )
{
  LDAP_Entry e = make_entry();
  ldap.add_entry(e);
}

void Obj::update_in_ldap( LDAP_Wrap &ldap )
{
  LDAP_Entry e = make_entry();
  ldap.update_entry(e);
}

LDAP_Entry Obj::make_entry()
{
  LDAP_Entry e;

  // need to build dn first
  if ( dn.empty() )
    {
      if ( id.empty() || parentDN.empty() )
          throw( x_base( TPFX + "Can not figure out dn" ) );
      dn = ID_Attr + '=' + id + ", " + parentDN;
    }

  // dn
  e.dn = dn;

  // objectclass
  string top = "top";
  e[ Obj_Class_Attr ] = make_vector( top, CFAT_Obj, type );

  // id
  e[ ID_Attr ] = make_vector( id );

  // cn
  e[ CN_Attr ] = make_vector( cn );

  return e;
}

void Obj::init_from_entry( LDAP_Entry &e )
{
  // dn
  dn = e.dn;

  // ID
  id = extract_id_from_dn( dn );

  // parentDN
  parentDN = extract_parent_from_dn( dn );

  // cn
  if ( e.has_a_val( CN_Attr ) )
    cn = e.val( CN_Attr );
}

Obj *Obj::make_obj_from_entry( LDAP_Entry &e )
{
  Obj *ret = NULL;

  if ( !e.has_a_val( Obj_Class_Attr ) )
    throw (x_base( TPFX + "Object type not specified" ) );

  ret = make_obj( e[ Obj_Class_Attr ].back() );
  (*ret).init_from_entry( e );

  return ret;
}

Obj *Obj::make_obj( const string &obj_type )
{
  Obj *ret = NULL;

  if ( obj_type == Request_Obj )
    ret = new Request;
  else if ( obj_type == Job_Order_Obj )
    ret = new Job_Order;
  else if ( obj_type == Action_Obj )
    ret = new Action;
  else if ( obj_type == Folder_Obj )
    ret = new Folder;
  else if ( obj_type == Job_Obj )
    ret = new Job;
  else if ( obj_type == Agent_Obj )
    ret = new Agent;
  else if ( obj_type == Engine_Obj )
    ret = new Engine;
  else
    throw( x_base( TPFX + "Unknown object type: " + obj_type ) );

  return ret;
}
```

APPENDIX A
SOURCE CODE LISTING

```cpp
string Obj::print_url()
{
  string ret;

  // dn
  ret += "dn=";
  ret += url_encode( dn );

  // type
  ret += (string)"&type=" + type;

  // id
  ret += "&id=";
  ret += url_encode( id );

  // cn
  ret += "&cn=";
  ret += url_encode( cn );

  return ret;
}


//=============================
// Broken Implementation
//=============================
void Broken::init_from_entry( LDAP_Entry &e, string err )
{
  Obj::init_from_entry( e );
  error = err;
}

string Broken::print_url()
{
  string ret = Obj::print_url();
  ret += "&err=";
  ret += url_encode( error );
  return ret;
}


//=====================
// Job Implementation
//=====================
void Job::init_from_entry( LDAP_Entry &e )
{
  Obj::init_from_entry( e );

  if ( !e.peek( Command_Attr ))
    throw(x_base(TPFX+"No command attr in a job entry"));
  command = e.val( Command_Attr );

  if ( !e.has_a_val( AgentDN_Attr ))
    throw(x_base(TPFX+"No agent DN in a job entry"));
  agentDN = e.val( AgentDN_Attr );

  if ( !e.peek( Param_Attr ) )
    throw(x_base(TPFX+"No param attr in a job entry"));
  p_vec = e[ Param_Attr ];

  if ( !e.peek( Rval_Attr ) )
    throw(x_base(TPFX+"No rval attr in a job entry"));
  r_vec = e[ Rval_Attr ];
}

string Job::print_url()
{
  string ret = Obj::print_url();

  // param list
  for ( int i=0; i<p_vec.size(); i++ )
    ret += (string)"&param=" + url_encode( p_vec[i] );

  // rval list
  for ( int i=0; i<r_vec.size(); i++ )
    ret += (string)"&rval=" + url_encode( r_vec[i] );

  // command
  ret += (string)"&command=" + url_encode( command );

  // agentdn
  ret += (string)"&agentDN=" + url_encode( agentDN );

  return ret;
}

LDAP_Entry Job::make_entry()
{
  LDAP_Entry e = Obj::make_entry();
  e[ Param_Attr ] = p_vec;
  e[ Rval_Attr ] = r_vec;
  e[ AgentDN_Attr ] = make_vector( agentDN );
  e[ Command_Attr ] = make_vector( command );

  return e;
}

//========================
// Engine Implementation
//========================
```

APPENDIX A
SOURCE CODE LISTING

```cpp
void Engine::init_from_entry( LDAP_Entry &e )
{
  Obj::init_from_entry( e );
}

string Engine::print_url()
{
  string ret = Obj::print_url();
  return ret;
}

LDAP_Entry Engine::make_entry()
{
  LDAP_Entry e = Obj::make_entry();
  return e;
}
XXXXXXXXXX END /share/Kiki/WF/prod/util/Obj.cc XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Params.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/*************************************************************
 *
 *    $Id: Params.cc,v 1.40 1999/02/18 01:07:25 rt Exp $
 *
 *    Desc: Parameter handling (read, parse, make map, print, etc)
 *
 *************************************************************/
#include "WF.h"

//=========================
// Param implementation
//=========================
void Param::add_val( const string &name, const string &value )
{
  if ( peek( name ) )
    operator[]( name ).push_back( value );
  else
    {
      String_Vector tmp;
      tmp.push_back( value );
      operator[]( name ) = tmp;
    }
}

void Param::parse_assign_vector( String_Vector &v )
{
  string name;
  string value;

  for (int i=0; i<v.size(); i++)
    {
      split_assignment( v[i], name, value );
      add_val( name, value );
    }
}

void Param::parse_str( const string &s ) {
  string name;
  string value;
  String_Vector lines = split(s, '\n');
  for(int i=0; i<lines.size(); i++) {
    split_assignment( lines[i], name, value );
    add_val( name, value );
  }
}

String_Vector Param::make_assign_vector()
{
  String_Vector ret;

  for ( Param::iterator i=begin(); i!=end(); i++ )
    {
      int sz = i->second.size();
      for ( int j=0; j<sz; j++ )
        ret.push_back( i->first + '=' + i->second[j] );
    }

  return ret;
}

string Param::print_str()
{
  string ret;

  for ( iterator i = begin(); i != end(); i++ )
    {
      int sz = (*i).second.size();
      for ( int j=0; j<sz; j++ )
        ret += (*i).first + "=" + (*i).second[j] + "\n";
    }
  return ret;
}

// parse url-encoded string into name/value pairs
// store in param map
void Param::parse_url_enc_params(const string& str)
{
  string enc_name, enc_value;
  string tmp;
  string name, value;
```

67

APPENDIX A
SOURCE CODE LISTING

```cpp
   string::size_type beg;
   string::size_type i = 0;
   string::size_type sz = str.size();
   int len;
   string::size_type iter;

   while ( i < sz )
      {
         // beginning of parameter assignment
         beg = str.find_first_not_of( '&', i );

         // skip to end of str or next &
         i = str.find_first_of( '&', beg );

         // length of parameter assignment
         if ( i == string::npos )
             len = string::npos;
         else
             len = i - beg;

         // parameter assignment
         tmp = str.substr(beg, len);

         // get encoded name and value
         split_assignment( tmp, enc_name, enc_value );

         // decode
         name = url_decode(enc_name);
         value = url_decode(enc_value);

         // store in param map
         add_val( name , value );
      }
}

//==================================
// Config_Param implementation
//==================================
void Config_Param::init(int argc, char **argv)
{
   // Parse argv into params
   parse_argv_params( argc, argv );

   // Parse configs if specified
   if ( has_a_val( Config_File_Param ) )
      parse_config_files();

   // Set up quick access to some params
   set_up_globals();
}

void Config_Param::set_up_globals()
{
   if ( has_a_val( BindDN_Param ) )
      bindDN = val( BindDN_Param );
   if ( has_a_val( BindPW_Param ) )
      bindPW = val( BindPW_Param );
   if ( has_a_val( Server_Param ) )
      server = val( Server_Param );
   if ( has_a_val( Param_Param ) )
      user_map.parse_assign_vector( operator[]( Param_Param ) );
}

void Config_Param::parse_config_files()
{
   if ( !peek( Config_File_Param ) )
      return;

   String_Vector &cfgs = operator[]( Config_File_Param );
   int j;

   for ( int i=0; i<cfgs.size(); i++ )
      {
         // check that we have not read this config yet
         for ( j=0; j<i && cfgs[j] != cfgs[i]; j++ );
         if ( j<i ) continue;

         // '-' means read stdin
         if ( cfgs[i] == "-" )
            {
               parse_file( cin, cfgs[i] );
               continue;
            }

         // open config file
         ifstream cfg_file( cfgs[i].c_str() );

         if ( !cfg_file )
            throw( x_base( TPFX + "Unable to open config file " + cfgs[i] ) );

         // read config file
         parse_file( cfg_file, cfgs[i] );
      }
}

// read file, parse params, close file when done
void Config_Param::parse_file( istream &a_file, const string &file_name )
{
   string a_line;
   string::size_type i;
```

APPENDIX A
SOURCE CODE LISTING

```
    string name, value;

    while ( getline( a_file, a_line ) )
           {
                   // empty line
                   if ( a_line.empty() )
                           continue;

                   // skip comments
                   if ( a_line.at(0) == '#' )
                           continue;

                   // skip white space lines
                   if ( a_line.find_first_not_of( " \t\n\r" ) == string::npos )
                           continue;

                   // split into name/value
                   split_assignment( a_line, name, value );

                   // store
                   add_val( name, value );
           }

    if ( a_file.bad() )
           throw( x_base( TPFX + "Error reading config file " + file_name ) );
}

// Split args by '=' and add to parameter name to value map
void Config_Param::parse_argv_params( int argc, char **argv )
{
    int i;
    string p_name;
    string p_value;

    for ( i=1; i < argc; i++ )
           {
           split_assignment( argv[i], p_name, p_value );
           add_val( p_name, p_value );
           }
}
XXXXXXXXXX END /share/Kiki/WF/prod/util/Params.cc XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Req_Builder.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/
/*********************************************************************
 *
 *     $Id: Req_Builder.cc,v 1.20 1999/02/21 04:20:34 rt Exp $
 *
 *     Desc: Parse action and jobs, build req proto, and instanciate
 *
 *********************************************************************/


#include "WF.h"

void Req_Builder::build_protos( LDAP_Wrap &ldap, const string &a_dn )
{
    // Get the action
    Action action;
    action.dn = a_dn;
    action.init_from_ldap( ldap );

    // Get the jobs
    Job_Vec jv;
    for ( int i=0; i<action.script.size(); i++ ) {
        Job j;
        j.dn = action.script.line(i).job_dn();
        j.init_from_ldap( ldap );
        jv.push_back( j );
    }

    // validate action
    //validate_action( action, jv );

    // build req proto
    build_req_proto( action );

    // build job order protos
    for (int i=0; i < jv.size(); i++)
        build_jo_proto( jv[i] );
}

void Req_Builder::validate_action( Action &a, Job_Vec &jv ) {
    for ( int i=0; i<jv.size(); i++ ) {
        // job parameters
        String_Vector &pv = jv[i].p_vec;

        // parameter assignments for current scipt line
        Param &pm = a.script.line(i).p_map();
        // return parameter assignments
        Param &rm = a.script.line(i).r_map();

        // build action param map - for quick lookup
        Param am;
        for (int j=0; j<a.p_vec.size(); j++)
            am.add_val(a.p_vec[j], 1);

        // for each job parameter check it's passed a value
        for ( int j=0; j<pv.size(); j++ ) {
```

69

APPENDIX A
SOURCE CODE LISTING

```
        // there are parameter mappings for this parameter
        if ( pm.peek(pv[j]) ) {

            // should not happen
            if ( pm[pv[j]].size() <= 0 )
                throw(x_base(TPFX+"Invalid parameter assignment for param ["
                            +pv[j]+"]"));

            // check that each assignment token appears in action params
            for(int k=1; k<pm[pv[j]].size(); k++) {
                string &t = pm[pv[j]][k];
                if ( !am.peek(t) )
                    throw(x_base(TPFX+""));
            }
        }


        // no parameter assignments - has to be one of action params
        else {
            if ( !am.peek(pv[j]) )
                throw(x_base(TPFX+"Parameter ["+pv[j]+"] is not passed a value"));
        }
    }


    // check return values
}

void Req_Builder::build_req_proto( Action &action ) {
    // copy info
    req.actionDN = action.dn;
    p_vec = action.p_vec;
    req.cn = action.cn;
    req.parentDN = ID_Attr + '=' + Queue_Fldr + ", " + req.actionDN;
    req.script = action.script;

    // figure out which jobs are bg
    for (int i=0; i<req.script.size(); i++) {
        Job_State j;
        j.bg = req.script.line(i).bg();
        req.states.push_back( j );
    }
}
void Req_Builder::build_jo_proto( Job &job )
{
    Job_Order jo;
    jo.job_dn = job.dn;
    jo.command = job.command;
    jo.agent_dn = job.agentDN;
    jo.cn = job.cn;

    // init the p_map
    String_Vector empty_vec = make_empty_vector();
    for(int i=0; i<job.p_vec.size(); i++) {
        if ( !jo.p_vec[i].empty() )
            jo.p_map[job.p_vec[i]] = empty_vec;
    }

    // init the rvals map
    for(int i=0; i<job.r_vec.size(); i++) {
        if ( !jo.r_vec[i].empty() )
            jo.r_map[job.r_vec[i]] = empty_vec;
    }

    job_orders.push_back( jo );
}

Request Req_Builder::build_req( Request &in_req )
{
    Request r = req;
    r.id = in_req.id;
    r.dn = 'string'ID_Attr + '=' + r.id + ", " + req.parentDN;
    r.p_map = in_req.p_map;

    // check that all parameters have a value
    for (int i=0; i<p_vec.size(); i++)
    {
        if ( !r.p_map.peek( p_vec[i] ) )
            throw(x_base(TPFX+"Parameter ["+p_vec[i]+"] does not have value."));
    }

    return r;
}

Job_Order_Vec Req_Builder::build_jo_vec( Request &r )
{
    // copy prototypes
    Job_Order_Vec new_orders = job_orders;

    for (int seq_num=0; seq_num<job_orders.size(); seq_num++)
    {
        // build job order id, dn, and parentDN
        Job_Order &jo = new_orders[seq_num];
        jo.id = r.id + "]" + numlstr( seq_num );
        jo.dn = ID_Attr + '=' + jo.id + ", " + r.dn;
        jo.parentDN = r.dn;
        jo.action_dn = r.actionDN;
    }

    return new_orders;
}
```

APPENDIX A
SOURCE CODE LISTING

```cpp
void Req_Builder::prep_jo( Request &r, Job_Order &jo, int n ) {
  Param &am = r.script.line(n).p_map(); // param assignments
  Param &rm = r.p_map; // request param map
  Param &jm = jo.p_map; // job order param map

  // figure out parameter values
  for ( Param::iterator i=jm.begin(); i!=jm.and(); i++ ) {
    string &name = i->first;
    i->second.clear();

    // there is a parameter assignment
    if ( am.has_a_val(name) ) {
      i->second.push_back( am.val(name) );
      replace_markers( i->second[0], rm );
    }

    // no param assignment - pull value from request
    else {
      if ( !rm.has_a_val(name) )
        throw(x_base(TPFX+"No value for param ["+name+"]"));

      i->second.push_back( rm.val(name) );
    }
  }

  // replace param markers by param values in command
  replace_markers( jo.command, jm );
}

void Req_Builder::pull_ret_vals( Request &r, Job_Order &jo, int n ) {
  Param &am = r.script.line(n).r_map(); // param assignments
  Param &rm = r.p_map; // request param map
  Param &jm = jo.r_map; // job order param map

  for(Param::iterator i=rm.begin(); i!=rm.and(); i++) {
    string &name = i->first;

    if ( !rm.peek(name) )
      throw(x_base(TPFX+"No param ["+name+
                    "] in request to assign return value to."));

    if ( i->second.size() <= 0 )
      throw(x_base(TPFX+"No return value for param ["+name+"]"));

    rm[name].clear();
    rm.add_val(name, i->second[0]);
    replace_markers(rm.val(name), jm);
  }
}

void Req_Builder::replace_markers( string &s, Param &m )
{
  string name;
  string::size_type beg=0;
  int len;

  while ( found_marker( s, name, beg, len ) )
  {
    // did not find parameter in the map
    if ( !m.peek( name ) )
      throw( x_base( TPFX + "Unknown parameter: " + name ) );

    // substitute marker by param value
    s.replace( beg, len, m.val( name ) );

    // advance past the marker just replaced
    beg += m.val( name ).size();
  }
}
// XXXXXXXXXX END /share/Kiki/WF/prod/util/Req_Builder.cc XXXXXXXXXX
// XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Request.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/
/*..........................................................................
 *
 *    $Id: Request.cc,v 1.19 1999/02/17 04:49:54 rt Exp $
 *
 *    Desc: Request obj implementation
 *
 *..........................................................................*/

#include "WF.h"

LDAP_Entry Request::make_entry()
{
  LDAP_Entry e = Obj::make_entry();

  // param
  e[ Param_Attr ] = p_map.make_assign_vector();

  // status
  e[ States_Attr ] = make_vector( encoded_states() );

  // pc
  e[ PC_Attr ] = make_vector( pc );

  // status
  e[ Status_Attr ] = make_vector( status );
```

APPENDIX A
SOURCE CODE LISTING

```cpp
  // log
  e[ Log_Attr ] = make_vector( log );

  // script
  e[ Script_Attr ] = make_vector( script.print_str() );

  // actionDN
  e[ ActionDN_Attr ] = make_vector( actionDN );

  return e;
}

string Request::encoded_states()
{
  string ret;
  for (int i=0; i<states.size(); i++)
      ret += num2str( states[i].status + 5*states[i].bg );
  return ret;
}

void Request::decode_states( const string &s )
{
  string::size_type i;
  int num;

  for ( i=0; i < s.size(); i++ )
    {
      num = s[i] - '0';
      Job_State js;
      js.bg = num >> 5;
      js.status = get_exec_status( num % 5 );
      states.push_back( js );
    }
}

void Request::init_from_entry( LDAP_Entry &e )
{
  Obj::init_from_entry( e );

  // actionDN
  if ( !e.has_a_val( ActionDN_Attr ) )
    throw(a_base(TPFX+"No actionDN attr"));
  actionDN = e.val(ActionDN_Attr);

  // status
  if ( !e.has_a_val( Status_Attr ) )
    throw(a_base(TPFX+"No status attr"));
  status = get_exec_status( e.val( Status_Attr ) );

  // pc
  if ( !e.has_a_val( PC_Attr ) )
    throw(a_base(TPFX+"No pc attr"));
  pc = safe_atoi( e.val( PC_Attr ) );

  // job states
  if ( e.has_a_val( States_Attr ) )
    decode_states( e.val( States_Attr ) );

  // params
  if ( e.has_a_val( Param_Attr ) )
    p_map.parse_assign_vector( e[ Param_Attr ] );

  // log
  if ( e.has_a_val( Log_Attr ) )
    log = e.val( Log_Attr );

  // script
  if ( e.has_a_val( Script_Attr ) )
    script.init( e.val( Script_Attr ) );
}

void Request::init_from_url( string a_url ) {
  LDAP_Entry e;
  e.init_from_url( a_url );
  init_from_entry( e );

  // get the job states
  //  for ( int i=0; e.has_a_val((string)"jo"+num2str(i)); i++ )

}

string Request::print_url()
{
  string ret = Obj::print_url();

  // status
  ret += "&status=";
  ret += Exec_Status_Name[ status ];

  // pc
  ret += "&pc=";
  ret += num2str( pc );

  // param
  String_Vector tmp = p_map.make_assign_vector();
  for ( int i=0; i<tmp.size(); i++ )
    ret += (string)"&param=" + url_encode( tmp[i] );

  // states
```

APPENDIX A
SOURCE CODE LISTING

```
  for ( int i=0; i<states.size(); i++ )
    {
      ret += (string)"&jo" + num2str(i) + '=';
      ret += states[i].print_url();
    }

  // log
  ret += (string)"&log=" + url_encode( log );

  // actionDN
  ret += (string)"&actiondn=" + url_encode( actionDN );

  // script
  ret += (string)"&script=" + script.print_url();

  return ret;
}

void Request::generate_new_req( const string &actdn, Param & in_param,
                                LDAP_Wrap &ldap )

{
  // get action from ldap
  Action act;
  act.dn = actdn;
  act.init_from_ldap( ldap );

  // copy over values
  cn = act.cn;
  actionDN = act.dn;
  p_map = in_param;
  parentDN = ID_Attr+'='+In_Fldr+", "+actionDN;

  // check that all parameters have a value
  for (int i=0; i<act.p_vec.size(); i++)
    {
        if ( !p_map.peek( act.p_vec[i] ) )
            throw( x_base( TPFX + "Parameter [" +
                           act.p_vec[i] + "] does not have value." ) );
    }

  // finally generate new id
  id = get_unique_id();
}


XXXXXXXXXX END /share/Kiki/WF/prod/util/Request.cc XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Script.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/*********************************************************************
 *
 *      $Id: Script.cc,v 1.7 1999/02/15 08:16:30 rt Exp rt $
 *
 *      Desc: Parse action script
 *
 *********************************************************************/


#include "WF.h"

// split script, where lines are separated by ';' into lines
void Script::split_script(const string &s) {
  String_Vector sv = split( s, ';' );
  lines.clear();
  for (int i=0; i<sv.size(); i++) {
    Script_Line tmp(sv[i]);
    lines.push_back( tmp );
  }
}

void Script::parse_all() {
  for(int i=0; i<size(); i++)
    lines[i].parse();
}

void Script_Line::parse() {
  if ( parsed )
    return;

  // reset everything
  job = "";
  bg_val = false;
  pmap.clear();
  rmap.clear();

  // figure out job dn
  string::size_type beg = l.find_first_not_of( " \t\n" );
  string::size_type end = l.find_first_of( "(" );
  string::size_type len;
  if ( end != string::npos )
    len = end - beg;
  else
    len = string::npos;
  job = l.substr( beg, len );

  // get p_map
  // first set of () is param map
  if ( end != string::npos ) {
    beg = end + 1;
    if ( beg >= l.size() )
```

APPENDIX A
SOURCE CODE LISTING

```cpp
      throw(x_base(TPFX+"Invalid script line syntax: "+l));
    end = l.find_first_of( ')', beg );
    if ( end == string::npos )
      throw(x_base(TPFX+"Invalid script line syntax: "+l));
    len = end - beg;
    string tmp = l.substr( beg, len );
    parse_map( tmp, pmap );
  }

  // get r_map
  // second set of () is return map
  if ( end != string::npos
       && (beg=l.find_first_of('(', end)) != string::npos )
    {
      beg++;
      if ( beg >= l.size() )
        throw(x_base(TPFX+"Invalid script line syntax: "+l));
      end = l.find_first_of( ')', beg );
      if ( end == string::npos )
        throw(x_base(TPFX+"Invalid script line syntax: "+l));
      len = end - beg;
      string tmp = l.substr( beg, len );
      parse_map( tmp, rmap );
    }

  // should it run in background?
  if ( end != string::npos )
    bg_val =  l.find( '&', end ) != string::npos;

  parsed = true;
}

void Script_Line::parse_map( const string &s, Param &m ) {
  // first get all the parameter mappings
  String_Vector sv = split( s, ',' );
  n.parse_assign_vector( sv );

  // now go through all assignments and extract references to other params
  for ( Param::iterator i=m.begin(); i!=m.end(); i++ ) {
    int sz = i->second.size();
    if ( sz <= 0 || sz > 1 )
      throw x_base(TPFX+"Invalid script syntax for param: "+i->first);
    extract_refs(i->second[0], i->second);
  }
}

void Script_Line::extract_refs( const string s, String_Vector &v ) {
  string m;
  string::size_type i = 0;
  int len = 0;
  while ( found_marker(s, m, i, len) ) {
    v.push_back(m);
    i += len;
  }
}

XXXXXXXXXX END /share/Kiki/WF/prod/util/Script.cc XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Util.cc XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/******************************************************************
 *
 *    $Id: Util.cc,v 1.43 1999/02/18 01:01:11 rt Exp $
 *
 *    Desc: Generic utility functions used throughout
 *
 ******************************************************************/

#include "WF.h"

// for strtol needed by url_decode
#include <stdlib.h>

// for sprintf needed by num2str
#include <stdio.h>

// for gethostid and getpid needed by get_unique_id
#include <unistd.h>

// for time needed by get_unique_id, time_stamp, now
#include <time.h>
// for some reason we don't pick this up from time.h
//extern char *ctime_r(const time_t *clock, char *buf, int buflen);

// for isxdigit needed by url_decode
// for tolower needed by downcase
#include <ctype.h>

// for gethostbyname and struct hostent used by get_my_ip
#include <netdb.h>
#include <arpa/inet.h>

// Decode url-encoded string
string url_decode ( const string& encoded )
{
  string tmp(encoded);
  string digits = "";
  char c;
  string::size_type i = 0;
```

APPENDIX A
SOURCE CODE LISTING

```cpp
// Replace + by space
for( i=0; i < tmp.size(); i++ )
    if ( tmp[i] == '+' )
        tmp[i] = ' ';

// The for loop is designed to go through the string only once
// replacing %DD by char corresponding to DD in hex.
for ( i=0; i+2 < tmp.size(); i++ )
{
        // not a %dd
        if ( tmp[i] != '%' )
            continue;

        // one of the two chars after ' is not a hex digit
        if ( !isxdigit(tmp[i+1]) || !isxdigit(tmp[i+2]) )
            continue;

        // found %dd - convert it
        digits = tmp.substr( i+1, 2 );
        c = (char)strtol( digits.c_str(), NULL, 16 );

        // replace 3 chars "%dd" by one char c
        tmp.replace( i, 3, 1, c);
    }

    return tmp;
}

// url encode a string
string url_encode( const string &decoded )
{
    string ret = decoded;
    string::size_type i = 0;
    string tmp;

    for( i=0; i < ret.size(); i++ )
    {
        // alfa-numeric need not be changed
        if ( isalnum( ret[i] ) )
            continue;

        // replace space by +
        if ( ret[i] == ' ' )
        {
            ret[i] = '+';
            continue;
        }

        // all weird chars need to be replaced by %DD
        tmp = "%";
        if ( (int)ret[i] < 16 )
            tmp += "0";
        tmp += num2str( (int)ret[i], 16 );
        ret.replace( i, 1, tmp );

        // advance two extra for the DD
        i += 2;
    }

    return ret;
}

// Print error and exit
void die( const string& errmsg )
{
    cerr << errmsg << endl;
    exit( -1 );
}

// Convert integer to a string
string num2str( int n , int base )
{
    char buf[12];

    if ( base == 16 )
        sprintf( buf, "%x", n );
    else
        sprintf( buf, "%d", n );
    return (string)buf;
}

// Convert long to a string
string num2str( long n, int base )
{
    char buf[12];
    if ( base == 16 )
        sprintf( buf, "%x", n );
    else
        sprintf( buf, "%d", n );
    return (string)buf;
}

// Get unique identifier: t(current_time)p(pid)h(hostid)
// Note it is not unique in multi-thread environment
string get_unique_id()
{
    time_t cur_time = now();
    pid_t my_pid = getpid();
```

75

APPENDIX A
SOURCE CODE LISTING

```cpp
   return (string)"t" + num2str(cur_time) + "p" +
      num2str(my_pid) + "h" + get_my_ip();
}

string get_my_ip() {
   char buf[256];
   int ret;
   buf[255] = '\0';
   ret = gethostname(buf, 256);
   if ( ret != 0 || buf[255] != '\0' )
      throw(x_base(TPFX+"Can not get local host name"));

   struct hostent *e = gethostbyname(buf);
   if ( e == NULL )
      throw(x_net(TPFX+"gethostbyname: "));

   string ipaddr(iner_ntoa(*((struct in_addr *)(e->h_addr_list[0]))));
   return ipaddr;
}

char *dup_c_str( const char *s )
{
   if ( !s )
      return NULL;

   char *ret = strdup( s );

   if ( !ret )
      throw( x_base( TPFX + "Insufficient memory" ) );

   return ret;
}

void split_assignment( const string& a, string &name, string &value )
{
   String_Vector sv = split(a, '=', 2);

   // did not find name
   if ( sv[0].empty() )
      throw( x_base( TPFX + "Invalid parameter assignment: " + a ) );

   // figure out name and value
   name = sv[0];
   value = sv[1];
}

// split string into array of strings by separator sep
// skip white space, if num > 0 returns vector of size num
// pads with "" if too few tokens found, or puts unsplit
// remainder in last element if too many tokens found
String_Vector split( const string &s, char sep, int num )
{
   // result
   String_Vector ret;

   // skip white space
   string skip = " \t\n\r";

   string::size_type beg = 0;
   string::size_type i = 0;
   string::size_type sz = s.size();
   int len;
   string tmp;
   int max;

   // exact vector length specified
   if ( num > 0 )
      max = num - 1;
   else
      max = sz;

   for ( i=0; i < sz && ret.size() < max; i++ ) {
      // beginning of token
      beg = s.find_first_not_of( skip, i );

      // nothing interesting found
      if ( beg == string::npos )
         break;

      // skip to end of str or next separator
      i = s.find_first_of( sep, beg );

      // length of token
      if ( i == string::npos )
         len = string::npos;
      else
         len = i - beg;

      // token
      tmp = s.substr(beg, len);
      ret.push_back( tmp );

      // done - exit loop
      if ( i == string::npos )
         break;
   }

   // still some unsplit content left - give it all in last element
   if ( i < sz ) {
      // skip white space
```

APPENDIX A
SOURCE CODE LISTING

```
    beg = s.find_first_not_of( skip, i );

    // ok, found something
    if ( beg != string::npos ) {
      tmp = s.substr( beg );
      ret.push_back( tmp );
    }
  }

  // pad return vector with "" if exact length specified
  if ( num > 0 )
    for(int j = num - ret.size(); j; j--)
      ret.push_back("");

  return ret;
}

string downcase( const char *str )
{
    string ret( str );

    string::size_type i=0;
    string::size_type sz = ret.size();

    for(i=0; i<sz; i++)
        ret[i] = tolower( ret[i] );

    return ret;
}

// finds marker <marker> in string s
// sets beg to beginning
// sets len to full length (including < and >)
// returns true if found, false otherwise
bool found_marker( const string &s, string &marker,
                   string::size_type &beg, int &len )
{
    string::size_type i = beg;

    // end of string - nothing to look for
    if ( beg > s.size() )
      return false;

    // beginning of param marker
    beg = s.find( "%<", i );

    // did not find "%<" - done
    if ( beg == string::npos )
      return false;

    // end of param marker
    i = s.find_first_of( '>', beg );

    // did not find closing '>' - done
    if ( i == string::npos )
      return false;

    // length of param marker
    len = i - beg + 1;

    // parameter name
    // +2 skips "%<" and -3 accounts for "%<" and closing ">"
    marker = s.substr(beg+2, len-3);

    return true;
}

// return string representation of current time
string time_stamp()
{
    time_t t = now();
    char buf[26];
    char *t_str = ctime_r( &t, buf, 26 );

    if ( !t_str )
      throw( x_sys( TPFX + "ctime_r: " ) );

    // 24 is to avoid the newline at the end
    string ret;
    ret.assign( buf, 24 );
    return ret;
}

// safe current time
time_t now()
{
    time_t t = time( NULL );
    if ( t < 0 )
      throw( x_sys( TPFX + "time: " ) );
    return t;
}

string extract_id_from_dn( const string & a_dn )
{
    string::size_type end = a_dn.find_first_of( ',' );
    string::size_type beg = a_dn.find_first_of( '=' );

    if ( end == string::npos || beg >= end )
      throw ( x_base( TPFX + "Malformed dn: " + a_dn ) );
```

APPENDIX A
SOURCE CODE LISTING

```
    string ret = a_dn.substr( beg+1, end - beg - 1 );
    return ret;
}

string extract_parent_from_dn( const string & a_dn )
{
    // end of rdn
    string::size_type end = a_dn.find_first_of( ',' );
    if ( end == string::npos )
        throw ( x_base( TPFX + "Malformed dn: " + a_dn ) );

    // beginning of parent dn
    string::size_type beg = a_dn.find_first_not_of( " \t\n,", end );
    if ( beg == string::npos )
        throw ( x_base( TPFX + "Malformed dn: " + a_dn ) );

    // extract parent
    string ret = a_dn.substr( beg );
    return ret;
}

String_Vector make_vector( const string &s )
{
    String_Vector ret;
    ret.push_back( s );
    return ret;
}

String_Vector make_vector( int i )
{
    string x = num2str(i);
    return make_vector( x );
}

String_Vector make_vector( const string &s1, const string &s2 )
{
    String_Vector ret;
    ret.push_back( s1 );
    ret.push_back( s2 );
    return ret;
}

String_Vector make_vector( const string &s1, const string &s2,
                           const string &s3 )
{
    String_Vector ret;
    ret.push_back( s1 );
    ret.push_back( s2 );
    ret.push_back( s3 );
    return ret;
}

int safe_atoi( const char *s ) {
    if ( s == NULL )
        throw(x_base(TPFX+"NULL string passed"));
    char *end = NULL;
    int i = strtol( s , &end, 10);
    if ( end == NULL || *end != '\0' )
        throw(x_sys(TPFX+"strtol: "));
    return i;
}

Cmd_Enum get_cmd_enum( int i ) {
    if ( i < 0 || i >= Cmd_Enum_Size )
        throw(x_base(TPFX+"Invalid Cmd_Enum value"));
    return (Cmd_Enum)i;
}

Exec_Status get_exec_status( int i ) {
    if ( i < 0 || i >= Exec_Status_Size )
        throw(x_base(TPFX+"Invalid Exec_Status value"));
    return (Exec_Status)i;
}
XXXXXXXXXX END /share/Kiki/WF/prod/util/Util.cc XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Action.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/*******************************************************************
 *
 *      $Id: Action.h,v 1.17 1999/02/17 05:18:37 rt Exp $
 *
 *      Desc: Action class
 *
 *******************************************************************/

class Action : public virtual Obj
{
public:
    string queueDN;
    string formURL;
    Script script;
    String_Vector p_vec;

    Action() : Obj( Action_Obj ) {};
    ~Action(){}

    void init_from_entry( LDAP_Entry &e );
    string print_util();
    LDAP_Entry make_entry();
    void add_to_ldap( LDAP_Wrap &ldap );
```

APPENDIX A
SOURCE CODE LISTING

```
};
XXXXXXXXXX END /share/Kiki/WF/prod/util/Action.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Exception.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/***********************************************************************
 *
 *     $Id: Exception.h,v 1.11 1999/02/14 02:46:28 rt Exp $
 *
 *     Desc: Exception handling
 *
 ***********************************************************************/


// Throw prefix to record file, function, and line number in error string
#define TPFX (string)"In "+__FILE__+" "+__FUNCTION__+"["+ num2str(__LINE__)+"]: "


// Base exception class
class x_base
{
public:
    int err;
    string msg;

    inline x_base( string in_msg = "", int in_err = 0 )
        : err( in_err ), msg( in_msg )){}
    inline x_base( const x_base &in_x )
        : err( in_x.err ), msg( in_x.msg ) {}
};


// Exception caused by system error (makes use of errno)
class x_sys : public x_base
{
public:
    x_sys( string u_msg = "" );
};

// Network exceptions
class x_net : public x_sys {
public:
    x_net( string u_msg = "" );
};

// LDAP related exceptions
class x_ldap : public x_base
{
public:
    inline x_ldap( int stat, string u_msg = "" )
        : x_base( u_msg + ldap_err2string( stat ), stat ) {}
};

XXXXXXXXXX END /share/Kiki/WF/prod/util/Exception.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Job_Order.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/***********************************************************************
 *
 *     $Id: Job_Order.h,v 1.11 1999/02/17 20:10:42 rt Exp $
 *
 *     Desc: Job_Order implementation
 *
 ***********************************************************************/


class Job_Order : public virtual Obj {
public:
    Job_Order() : status( Hold_Status ), Obj( Job_Order_Obj ) {}
    ~Job_Order(){}

    void init_from_entry( LDAP_Entry &e );
    LDAP_Entry make_entry();
    string print_url();

    Exec_Status status;
    string command;
    string job_dn;
    string agent_dn;
    string action_dn;
    string log;
    Param p_map;
    Param r_map;
};

typedef vector < Job_Order > Job_Order_Vec;
XXXXXXXXXX END /share/Kiki/WF/prod/util/Job_Order.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Job_State.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/***********************************************************************
 *
 *     $Id: Job_State.h,v 1.1 1998/12/16 00:41:01 rt Exp $
 *
 *     Desc: Job_State class
 *
 ***********************************************************************/
```

APPENDIX A
SOURCE CODE LISTING

```
class Job_State {
public:
  Job_State() : bq( false ), status( Runnable_Status ){};
  Job_State( const Job_State &j ) : bq( j.bq ), status( j.status ) {};
  ~Job_State(){};
  bool bq;
  int status;

  string print_url();
};

typedef vector < Job_State > Job_State_Vec;
XXXXXXXXXX END /share/Kiki/WF/prod/util/Job_State.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/LDAP_Entry.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/************************************************************************
 *
 *    $Id: LDAP_Entry.h,v 1.10 1999/01/18 20:55:02 rt Exp $
 *
 *    Desc: LDAP_Entry is map of attrs to their values - used to pass
 *          information about records in LDAP
 *
 ************************************************************************/


class LDAP_Entry : public virtual Param {
  public:
   // Full entry dn
   string dn;

   // Constructors and destructors
   LDAP_Entry(){}
   ~LDAP_Entry(){}
   LDAP_Entry(LDAP *ld, LDAPMessage *e) { init( ld, e ); }
   void init( LDAP *ld, LDAPMessage *e );
   void init_from_url( const string &url );

   // print LDIF format
   string print_str();

  protected:
   String_Vector parse_values (LDAP *ld, LDAPMessage *e, char *a);
};
XXXXXXXXXX END /share/Kiki/WF/prod/util/LDAP_Entry.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/LDAP_Wrap.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/************************************************************************
 *
 *    $Id: LDAP_Wrap.h,v 1.29 1999/01/31 08:D2:DD rt Exp $
 *
 *    Desc: LDAP_Wrap talks to LDAP API library
 *
 ************************************************************************/

class LDAP_Wrap
{
 protected:
   // connection handle
   LDAP *ld;

   // time to sleep (in sec) before reconnecting
   // if connection is lost
   int interval;

   // bind parameters
   string bind_dn;
   string bind_pw;
   string host;

 public:

   // low level - do the reconnecting if needed
   LDAP_Entry_Vec search( const string &base, int scope,
                          const string &filter = Default_Filter );
   void add( const string &a_dn, LDAPMod_NTA &mods );
   void modify( const string &a_dn, LDAPMod_NTA &mods );
   void remove( const string &a_dn );
   void disconnect();
   void connect() { bind(); }


   // constructors and destructors
   LDAP_Wrap( const string &a_host,
              const string &bindDN,
              const string &bindPW,
              int an_interval = -1 ) // -1 means no reconnect
     : interval( an_interval ),
       ld( NULL ),
       host( a_host ),
       bind_dn( bindDN ),
       bind_pw( bindPW ) {}

   LDAP_Wrap( const string &a_host,
              int an_interval = -1 ) // -1 means no reconnect
     : interval( an_interval ),
       ld( NULL ),
       host( a_host ){}
```

80

APPENDIX A
SOURCE CODE LISTING

```
LDAP_Wrap()
  : interval( -1 ), ld( NULL )()

~LDAP_Wrap() { disconnect(); }


  // higher level
  void modify_attr( const string &a_dn, const string &attr,
                    const string &val, int mod_type );
  void replace_attr( const string &a_dn, const string &attr,
                     const string &val )
    { modify_attr( a_dn, attr, val, LDAP_MOD_REPLACE ); }
  void replace_attr( const string &a_dn, attr, num2str( ival ), LDAP_MOD_REPLACE ); }
    { modify_attr( a_dn, attr, val, LDAP_MOD_REPLACE ); }
  void update_entry( LDAP_Entry &e );
  void add_attr( const string &a_dn, const string &attr, const string &val )
    { modify_attr( a_dn, attr, val, LDAP_MOD_ADD ); }
  void add_entry( LDAP_Entry &e );
  void remove_subtree( const string &a_dn );
  void move_subtree( const string &from, const string &to );
  bool exists( const string &a_dn );
  void prefs( const string &a_host, const string &bindDN,
              const string &bindPW, int an_interval );

protected:
  LDAP_Entry_Vec parse_res_chain(LDAPMessage *res);
  void bind();
};
XXXXXXXXXX END /share/Kiki/WF/prod/util/LDAP_Wrap.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/LDAP_related.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/***********************************************************************
 *
 *    $Id: LDAP_related.h,v 1.15 1998/12/14 19:11:15 rt Exp $
 *
 *    Desc: Aid in handling LDAPMod structures and
 *          NULL terminated char* arrays - to be used by LDAP_Wrap only
 *
 ***********************************************************************/

class Char_Star_NTA
{
public:
  Char_Star_NTA( int in_size = 10 );
  Char_Star_NTA( const Char_Star_NTA &c);
  Char_Star_NTA( String_Vector &v );
  ~Char_Star_NTA();
  void push_back( const char *elt );
  void push_back( const string &s ) { push_back( s.c_str() ); }

  operator char **();

  static void destroy_css( char **c );
  static char **dup_css( char **c );

protected:
  int size;
  int last;
  char **array;

  void extend();
};


class LDAPMod_NTA
{
public:
  LDAPMod_NTA( int in_size = 10 );
  LDAPMod_NTA( const LDAPMod_NTA &c);
  ~LDAPMod_NTA();

  operator LDAPMod **();
  void push_back( int op, const char *type, char **values );
  void push_back( int op, const string &type, char ** values )
    { push_back( op, type.c_str(), values ); }
protected:
  int size;
  int last;
  LDAPMod **array;

  void extend();
  LDAPMod *LDAPMod_dup( const LDAPMod *m );
  void LDAPMod_destroy( LDAPMod *m );
};

XXXXXXXXXX END /share/Kiki/WF/prod/util/LDAP_related.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Obj.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/***********************************************************************
 *
 *    $Id: Obj.h,v 1.16 1999/02/17 05:21:07 rt Exp $
 *
 *    Desc: Class Obj is parent to all objects
 *
 ***********************************************************************/
```

APPENDIX A
SOURCE CODE LISTING

```
class Obj {
public:
  string dn;
  string id;
  string type;
  string parentDN;
  string cn;

  Obj( const string &a_type ) : type( a_type ) {}


  virtual string print_url();
  virtual string print_ldif() { throw(x_base(TPFX+"Function not defined")); }
  virtual void init_from_ldap( LDAP_Wrap &ldap );
  virtual void init_from_url( string a_url );
  virtual void init_from_entry( LDAP_Entry &e );
  virtual void add_to_ldap( LDAP_Wrap &ldap );
  virtual void update_in_ldap( LDAP_Wrap &ldap );
  virtual LDAP_Entry make_entry();

  static Obj *make_obj( const string &a_type );
  static Obj *make_obj_from_entry( LDAP_Entry &e );
};

class Folder : public virtual Obj {
public:
  Folder() : Obj( Folder_Obj ) {}
  ~Folder() {}
};

class Agent : public virtual Obj {
public:
  Agent() : Obj( Agent_Obj ) {}
  ~Agent() {}
};

class Engine : public virtual Obj {
public:
  Engine() : Obj( Engine_Obj ){}
  ~Engine(){}
  void init_from_entry( LDAP_Entry &e );
  LDAP_Entry make_entry();
  string print_url();
};

class Broken : public virtual Obj {
public:
  string error;

  Broken() : Obj( Broken_Obj ) {}
  ~Broken(){}
  void init_from_entry( LDAP_Entry &e, string err = "" );
  string print_url();
private:
  void add_to_ldap( LDAP_Wrap &ldap )
    { throw(x_base(TPFX+"Function not defined")); }
};

class Job : public virtual Obj {
public:
  string agentDN;
  string command;
  String_Vector p_vec;
  String_Vector r_vec;

  Job() : Obj( Job_Obj ){}
  ~Job(){}
  void init_from_entry( LDAP_Entry &e );
  LDAP_Entry make_entry();
  string print_url();
};

typedef vector < Job > Job_Vec;
XXXXXXXXXX END /share/Kiki/WF/prod/util/Obj.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/otil/Params.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/*********************************************************************
 *
 *      $Id: Params.h,v 1.26 1999/02/18 01:09:02 rt Exp $
 *
 *      Desc: parameter handling (reading, parsing, making map, printing, etc)
 *
 *********************************************************************/

// Generic parameter map
class Param: public virtual Str_To_StrVec_Map {
public:
  // operator[] returns a vector of values mapped to the key
  // use val to retrieve one value only

  // get value number i
  virtual string &val( const string &a_key, int i=0 )
    { return operator[]( a_key )[i]; }

  // check if key is in map
  virtual bool peek( const string &a_key )
    { return ( find( a_key ) != end() ); }
```

APPENDIX A
SOURCE CODE LISTING

```cpp
  // check if key has a non-empty first value
  virtual bool has_a_val( const string &a_key )
    {return peek( a_key )&&!operator[]( a_key ).empty()&&!val( a_key ).empty();}

  // print name/value pairs
  virtual string print_str();

  // store name/value pairs from a string
  virtual void parse_str( const string &s );

  // store name/value pairs from vector
  virtual void parse_assign_vector( String_Vector &v );

  // make a vector of name=val pairs
  virtual String_Vector make_assign_vector();

  // store url-enc name/value pairs
  virtual void parse_url_enc_params( const string &encoded );

  // add a new value for a key
  virtual void add_val( const string &name, const string &value );
  virtual void add_val( const string &name, int value )
    { string strval = num2str(value); add_val( name, strval ); }
};


// Configuration parameters
class Config_Param : public virtual Param {
public:
  // user parameters (if any)
  Param user_map;

  // global vars
  string bindDN;
  string bindPW;
  string server;

  // Constructors and destructors
  Config_Param() ()
  ~Config_Param() ()
  Config_Param(int argc, char **argv) { init( argc, argv ); }
  void init( int argc, char ** argv );

protected:
  void parse_argv_params( int argc, char **argv);
  void parse_file( istream &a_file, const string &file_name );
  void parse_config_files();
  void set_up_globals();
};
XXXXXXXXXX END /share/Kiki/WF/prod/util/Params.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Req_Builder.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/****************************************************************************
 *
 *    $Id: Req_Builder.h,v 1.10 1999/02/17 01:53:21 rt Exp $
 *
 *    Desc: Parse action and jobs, build req proto, and instanciate
 *
 ****************************************************************************/

class Req_Builder
{
public:
  Req_Builder(){};
  ~Req_Builder(){};

  String_Vector p_vec;
  Request req;
  Job_Order_Vec job_orders;

  void build_protos( LDAP_Wrap &ldap, const string &a_dn );
  Request build_req( Request &in_req );
  Job_Order_Vec build_jo_vec( Request &r );
  static void validate_action( Action &a, Job_Vec &jv );
  static void prep_jo( Request &r, Job_Order &jo, int n );
  static void pull_ret_vals( Request &r, Job_Order &jo, int n );

protected:
  void build_req_proto( Action &action );
  void build_jo_proto( Job &job );
  static void replace_markers( string &s, Param &m );
};
XXXXXXXXXX END /share/Kiki/WF/prod/util/Req_Builder.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Request.h XXXXXXXXXX
/*-*- Mode: C++ -*-*/

/****************************************************************************
 *
 *    $Id: Request.h,v 1.12 1999/02/17 02:08:43 rt Exp $
 *
 *    Desc: Request class implementation
 *
 ****************************************************************************/

class Request : public virtual Obj {
public:
```

APPENDIX A
SOURCE CODE LISTING

```
//---------
// Members
//---------
// status of request
Exec_Status status;
// log of request
string log;
// current instruction
int pc;
// vector of job order states
Job_State_Vec states;
// vector of parameter names
String_Vector p_vec;
// parameter map
Param p_map;
// actionDN
string actionDN;
// script
Script script;

//------------------------------
// constructors and destructors
//------------------------------
Request() : status( Hold_Status ), pc( 0 ), Obj( Request_Obj ) {}
~Request(){}

//----------------------
// various init functions
//----------------------
void init_from_entry( LDAP_Entry &e );
void init_from_url( string a_url );
void generate_new_req( const string &actdn, Param & in_param,
                       LDAP_Wrap &ldap );

//---------------------------
// various output functions
//---------------------------
string print_url();
LDAP_Entry make_entry();

//------------------
// other functions
//------------------
void write_to_log( const string &s )
   { log += time_stamp() + ' ' + s + '\n'; }

protected:
  void decode_states( const string &a );
  string encoded_states();
};
XXXXXXXXXX END /share/Kiki/WF/prod/util/Request.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Script.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/
/**********************************************************************
 *
 *     $Id: Script.h,v 1.9 1999/02/21 01:33:50 rt Exp $
 *
 *     Desc: Parse action script
 *
 **********************************************************************/

class Script_Line {
public:
  Script_Line(): parsed(false) {};
  Script_Line(const string &a_line) : l(a_line), parsed(false) {};
  ~Script_Line(){};

  string &line(){return l;};
  string &job_dn(){parse(); return job;};
  Param &p_map(){parse(); return pmap;};
  Param &r_map(){parse(); return rmap;};
  bool bg(){parse(); return bg_val;};
  void parse();

protected:
  string l;
  string job;
  Param pmap;
  Param rmap;
  bool bg_val;
  bool parsed;

  void parse_map( const string &s, Param &m );
  void extract_refs( const string s, String_Vector &v );
};

typedef vector < Script_Line > Script_Line_Vec;

class Script
{
public:
  Script(){};
  Script(const string &s) {init(s);};
  ~Script(){};

  void init(const string &s){script=s; split_script(s);};
  void parse_all();
  Script_Line &line(int n){out(n); return lines[n];};
```

APPENDIX A
SOURCE CODE LISTING

```
  int size(){return lines.size();};
  string print_str(){return script;};
  string print_url(){string str=print_str(); return url_encode(str);};

protected:
  void cut(int n){if (n>=size()) throw(x_base(TPFX+"Line number too large"));}
  Script_Line_Vec lines;
  string script;
  void split_script(const string &s);
};
XXXXXXXXXX END /share/Kiki/WF/prod/util/Script.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Util.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/**********************************************************************
 *
 *     $Id: Util.h,v 1.17 1999/02/18 01:01:40 rt Exp $
 *
 *     Desc: Generic utility functions used throughout
 *
 **********************************************************************/

string url_decode( const string& encoded );
string url_encode( const string &decoded );
void die( const string& errmsg );
string num2str( int n , int base = 10 );
string num2str( long n , int base = 10 );
string get_unique_id(){;}
char *dup_c_str( const char *s );
void split_assignment( const string &s, string &name, string &value );
String_Vector split( const string &s, char sep, int num=-1 );
string downcase( const string *str );
bool found_marker( const string &s, string &marker,
                          string::size_type &beg, int &len );

string time_stamp();
time_t now();
string extract_id_from_dn( const string &a_dn );
string extract_parent_from_dn( const string &a_dn );
String_Vector make_vector( const string &s );
String_Vector make_vector( int i );
String_Vector make_vector( const string &s1, const string &s2 );
String_Vector make_vector( const string &s1, const string &s2,
                           const string &s3 );
inline String_Vector make_empty_vector() {return make_vector("");};
int safe_atoi( const char *s );
inline int safe_atoi( const string &s ) { return safe_atoi( s.c_str() ); };
Cmd_Enum get_cmd_enum( int i );
inline Cmd_Enum get_cmd_enum( const string &s )
{return get_cmd_enum( safe_atoi(s) );};
inline Cmd_Enum get_cmd_enum( const char *s )
{return get_cmd_enum( safe_atoi(s) );};
Exec_Status get_exec_status( int i );
inline Exec_Status get_exec_status( const string &s )
{return get_exec_status(safe_atoi(s));};
inline Exec_Status get_exec_status( const char *s )
{return get_exec_status(safe_atoi(s));};
string get_my_ip();
XXXXXXXXXX END /share/Kiki/WF/prod/util/Util.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/WF.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/**********************************************************************
 *
 *     $Id: WF.h,v 1.50 1999/02/16 01:20:23 rt Exp $
 *
 *     Desc: Main header file for libutil - for inclusion by
 *           non-executable code
 *
 **********************************************************************/

// stdlib headers
#include <string>
#include <map>
#include <vector>
#include <fstream>
#include <iostream>

// ldap api header
#include "ldap.h"

// Exceptions
#include "Exception.h"

// Types
typedef map < string, string, less<string> > String_Map;
typedef vector < string > String_Vector;
typedef map < string, String_Vector, less< string > > Str_To_StrVec_Map;

// Exec_Status
typedef enum { Complete_Status=0, Hold_Status=1, Runnable_Status=2,
               Running_Status=3, Error_Status=4,
               Exec_Status_Size } Exec_Status;
extern char *Exec_Status_Name[];

// Cmd_Enum
typedef enum { Copy_Cmd=0, Move_Cmd=1, Del_Cmd=2, Cmd_Enum_Size } Cmd_Enum;
extern char *Cmd_Enum_Name[];

// LDAPMods_NTA and Char_Star_NTA
#include "LDAP_related.h"
```

APPENDIX A
SOURCE CODE LISTING

```cpp
// Constant and global var declarations
#include "WF_const.h"

// Utility functions
#include "Util.h"

// Parameter handling
#include "Params.h"

// Global vars
extern Config_Param config;

// LDAPMessage wrapper
#include "LDAP_Entry.h"

typedef vector < LDAP_Entry > LDAP_Entry_Vec;

// LDAP connection wrapper
#include "LDAP_Wrap.h"

// Objects
#include "Obj.h"
#include "Script.h"
#include "Job_State.h"
#include "Request.h"
#include "Job_Order.h"
#include "Action.h"

// Action and Job parsing, creation of Requests and Job Orders
#include "Req_Builder.h"
XXXXXXXXXX END /share/Kiki/WF/prod/util/WF.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/WF_const.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/
/*****************************************************************************
 *
 *     $Id: WF_const.h,v 1.67 1999/02/17 04:15:21 rt Exp $
 *
 *     Desc: Set up constants and declare global vars
 *
 *****************************************************************************/


/* Constants */

// Defaults
const int Log_Size_Default = 1024;
const int Job_Timeout_Default = 20;
const string Default_Filter = "objectclass=cpat";

/* Attribute names */
const string ID_Attr = "objid";
const string Form_Attr = "form";
const string URL_Attr = "formurl";
const string CN_Attr = "cn";
const string Desc_Attr = "description";
const string Param_Attr = "param";
const string Rval_Attr = "rval";
const string Command_Attr = "command";
const string Status_Attr = "status";
const string Log_Attr = "log";
const string Obj_Class_Attr = "objectclass";
const string JobDN_Attr = "jobdn";
const string ActionDN_Attr = "actiondn";
const string Script_Attr = "script";
const string AgentDN_Attr = "agentdn";
const string States_Attr = "jobstates";
const string PC_Attr = "pc";
const string DN_Attr = "dn";

/* Schema Object names */
const string Action_Obj = "action";
const string Request_Obj = "request";
const string Agent_Obj = "agent";
const string Job_Order_Obj = "jobOrder";
const string Job_Obj = "job";
const string Engine_Obj = "engine";
const string Folder_Obj = "folder";
const string Broken_Obj = "broken";
const string CPAT_Obj = "cpat";

/* Folders */
const string In_Fldr = "in";
const string Out_Fldr = "archive";
const string Queue_Fldr = "queue";

/* Parameter names */
const string Server_Param = "server";
const string BindDN_Param = "bdn";
const string BindPW_Param = "bpw";
const string Config_File_Param = "cfg";
const string Log_File_Param = "log";
const string Interval_Param = "interval";
const string Once_Param = "once";
const string Log_Size_Param = "log_size";
const string Job_Timeout_Param = "job_timeout";
const string Param_Param = "param";
const string ObjDN_Param = "odn";
const string ObjID_Param = "oid";
```

APPENDIX A
SOURCE CODE LISTING

```
const string Obj_Param = "obj";
const string Filter_Param = "filter";
const string Scope_Param = "scope";
const string New_Param = "new";
const string Engine_Param = "enginedn";
const string Agent_Param = "agentdn";
const string ServiceDN_Param = "servicedn";
const string Cmd_Param = "cmd";
const string TargetDN_Param = "tdn";
XXXXXXXXXX END /share/Kiki/WF/prod/util/WF_const.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/WF_ext.h XXXXXXXXXX
/*-*- Mode: C++; -*-*/

/******************************************************************************
 *
 *    $Id: WF_ext.h,v 1.12 1999/02/14 02:46:09 rt Exp $
 *
 *    Desc: For inclusion by external code that uses libutil
 *
 ******************************************************************************/

#include "WF.h"

Config_Param config;
char *Exec_Status_Name[] = { "COMPLETE", "HOLD", "RUNNABLE", "RUNNING",
                             "ERROR" };
char *Cmd_Enum_Name[] = { "copy", "move", "del" };
XXXXXXXXXX END /share/Kiki/WF/prod/util/WF_ext.h XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/cgi-bin/edit_object XXXXXXXXXX
#!/bin/sh

command="/share/Kiki/WF/prod/perl/edit_object"
name="edit_object"
tmp="/tmp/err_$name$$"
lib="/share/Depot/ldapsdk-30-SOLARIS-export-ssl/lib"

rm -f $tmp
cd /share/Kiki/WF/prod/perl
LD_LIBRARY_PATH=$lib
export LD_LIBRARY_PATH

( $command ) 2>$tmp

if [ $? -ne 0 ]; then
     echo
     echo '<pre>'
     echo Errors while executing $command
     echo
     if [ -f $tmp ]; then
          cat $tmp
     fi
     echo '</pre>'
fi

rm -f $tmp
XXXXXXXXXX END /share/Kiki/WF/prod/cgi-bin/edit_object XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/cgi-bin/get_status XXXXXXXXXX
#!/bin/sh

command="/share/Kiki/WF/prod/perl/get_status"
name="get_status"
tmp="/tmp/err_$name$$"
lib="/share/Depot/ldapsdk-30-SOLARIS-export-ssl/lib"

rm -f $tmp
cd /share/Kiki/WF/prod/perl
LD_LIBRARY_PATH=$lib
export LD_LIBRARY_PATH

( $command ) 2>$tmp

if [ $? -ne 0 ]; then
     echo
     echo '<pre>'
     echo Errors while executing $command
     echo
     if [ -f $tmp ]; then
          cat $tmp
     fi
     echo '</pre>'
fi

rm -f $tmp
XXXXXXXXXX END /share/Kiki/WF/prod/cgi-bin/get_status XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/cgi-bin/run_action XXXXXXXXXX
#!/bin/sh

command="/share/Kiki/WF/prod/perl/run_action"
name="run_action"
tmp="/tmp/err_$name$$"
lib="/share/Depot/ldapsdk-30-SOLARIS-export-ssl/lib"

rm -f $tmp
cd /share/Kiki/WF/prod/perl
LD_LIBRARY_PATH=$lib
export LD_LIBRARY_PATH

( $command ) 2>$tmp

if [ $? -ne 0 ]; then
```

APPENDIX A
SOURCE CODE LISTING

```
    echo
    echo '<pre>'
    echo Errors while executing $command
    echo
    if [ -f $tmp ]; then
        cat $tmp

    fi
    echo '</pre>'
fi

rm -f $tmp
XXXXXXXXX END /share/Kiki/WF/prod/cgi-bin/run_action XXXXXXXXX
XXXXXXXXX BEGIN /share/Kiki/WF/prod/perl/CPAT.pm XXXXXXXXX
#===============================================================
#
#  $Id: CPAT.pm,v 1.23 1999/08/24 23:22:42 rt Exp $
#
# Desc: library for using C-code and gui
#
#===============================================================
package CPAT;
require Exporter;
@ISA = qw(Exporter);
@EXPORT = qw( $CPATDIR $CPATCFG Get_Obj %Scope Run_Act Put_Obj $BaseDN );
use CGI;
use vars qw/ $getobj $get_obj $runact $run_act %O /;

# Config vars
$CPATDIR='/share/Kiki/WF/prod';
$CPATTMP='/tmp';
$CPATCFG="$CPATDIR/syscfg";
$BaseDN = 'objid=TOP, o=NONE';
-Scope = ( base=>0, level=>1, tree=>2 );
$getobj="$CPATDIR/ui/get_obj";
$runact="$CPATDIR/ui/run_action";
$putobj="$CPATDIR/ui/update_obj";

# Helper vars
%O = ( scope=>'scope', odn=>'odn', filter=>'filter', conf=>'cfg',
        bdn=>'bdn', bpw=>'bpw', param=>'param' );
$get_obj="$getobj $O{conf}=$CPATCFG";
$run_act="$runact $O{conf}=$CPATCFG";
$put_obj="$putobj $O{conf}=$CPATCFG";

sub Get_Obj {
    my %p = @_ ;
    my $type = $p{type} || 'top';
    my $filter = "objectclass=$type";
    $filter = "(&($filter)(cn=$p{cn}))" if ($p{cn});
    $filter = "(&($filter)($p{filter}))" if ($p{filter});
    $filter = "(&($filter)(objid=$p{id}))" if ($p{id});
    $p{filter}=$filter;
    $p{dn} ||= $BaseDN;
    my $odn = "$O{odn}=$p{dn}";
    my $scope = exists $p{scope} ? $p{scope} : $$Scope{tree};
    $scope = "$O{scope}=$scope";
    $filter = "$O{filter}=$p{filter}";
    my (@ret, @lines);
    my $command = "$get_obj '$odn' '$scope' '$filter' 2>&1";
    @lines = `$command` ;

    # command failed
    if ( $? ) {
        my $err = "Command [ $command ] failed with status $? and output:\n@lines";
        $p{safe} && return $err;
        die($err);
    }

    # command succeeded
    grep { chomp; push @ret, new CGI($_); } @lines;
    return @ret;
}
sub Run_Act {
    my %p = @_;

    # param checks
    ($p{dn} ne '') || die('No action dn specified');
    ($p{bdn} ne '') || die('No bind dn specified');
    ($p{bpw} ne '') || die('No bind password specified');

    # prep command
    my $tmp = "$(CPATTMP)/run_act_$$";
    my $command = "$run_act $O{conf}=-- >$tmp 2>&1";
    system('/usr/bin/rm', '-f', $tmp);

    # run the command
    open(RUN, "| $command");
    print RUN "$O{odn}=$p{dn}\n";
    print RUN "$O{bdn}=$p{bdn}\n";
    print RUN "$O{bpw}=$p{bpw}\n";
    my @param = $p{act}->param('param');
    for (@param)
    {
        my $val = $p{"_$_"};
        print RUN "$O{param}=$_=$val\n";
    }
    close RUN;
    my $stat1 = $?;

    # read the output
```

88

APPENDIX A
SOURCE CODE LISTING

```perl
open(TMP, "<${CPATTMP}/run_act_$$");
my @lines = <TMP>;
close TMP;
my $stat2 = $?;
system('/usr/bin/rm', '-f', $tmp);

# Catch errors
$stat1 && die("Command { ${command} } failed with status $stat1:\n@lines");
$stat2 && die("Can not read output of { ${command} }");
chomp $lines[0];
return new CGI($lines[0]);
}

sub Put_Obj {
  my $p = @_;

  # param checks
# ($p{bdn} ne '') || die('No bind dn specified');
# ($p{bpw} ne '') || die('No bind password specified');
  ($p{obj}->param('dn') ne '') || die('No dn specified in object');
  my $url_obj = $p{obj}->query_string();

  # prep command
  my $tmp = "${CPATTMP}/put_obj_$$";
  my $command = "$put_obj $O{conf}=- >$tmp 2>&1";
  system('/usr/bin/rm', '-f', $tmp);

  # run the command
  open(RUN, "| $command");

# print RUN "$O{bdn}=$p{bdn}\n";
# print RUN "$O{bpw}=$p{bpw}\n";
  print RUN "naw=1\n" if $p{new};
  print RUN "obj=$url_obj\n";
  close RUN;
  my $stat1 = $?;

  # read the output
  open(TMP, "<$tmp");
  my @lines = <TMP>;
  close TMP;
  my $stat2 = $?;
  system('/usr/bin/rm', '-f', $tmp);

  # Catch errors
  $stat1 && die("Command { ${command} } failed with status $stat1:\n@lines");
  $stat2 && die("Can not read output of { $command }");
  chomp $lines[0];
  return new CGI($lines[0]);
}
XXXXXXXXXX END /share/Kiki/WF/prod/perl/CPAT.pm XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/perl/edit_object XXXXXXXXXX
#!/usr/bin/perl -w

#================================================
#
# $Id: edit_object,v 1.57 1999/02/02 17:46:55 root Exp $
#
# Desc: edit/create/delete/move objects (action, job, agent,
#       engine, folder)
#
#================================================
use strict;
use CPAT;
use CPAT::CGI;
use CPAT::Edit;
use CGI qw/:standard center/;

CGI_Act( %Edit_Functions );
exit(0);
XXXXXXXXXX END /share/Kiki/WF/prod/perl/edit_object XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/perl/get_status XXXXXXXXXX
#!/usr/bin/perl -w

#================================================
#
# $Id: get_status,v 1.25 1999/02/25 01:48:35 rt Exp $
#
# Desc: Browse request queues and check on
#       status of requests and job orders
#
#================================================

# setup
use strict;
use CPAT;
use CPAT::CGI;
use CGI qw/:standard center/;
use vars qw/ $title /;
$title = 'Get Status';
# Main
CGI_Act
  (
    unknown  => \&Output_Initial_Page,
    find     => \&Handle_Find,
    display  => \&Handle_Display,
    showreq  => \&Show_Req,
  );

exit(0);
```

89

APPENDIX A
SOURCE CODE LISTING

```perl
#============
# Functions
#============
sub Show_Req {
  my $dn = param('reqdn');
  Delete_all();
  &Display_Request($dn);
}

sub Get_Obj_Safe {
  my @ret=Get_Obj(safe=>1,@_);
  if (not ref($ret[0])) {
    ($ret[0] =~ /no such object/i) && return ();
    die($ret[0]);
  }
  return @ret;
}

sub Get_Req_Obj {
  my *p = @_ ;
  my $dn = $p{dn};
  my $req;

  ($req)=Get_Obj_Safe(dn=>$dn, scope=>$Scope{base}, type=>'request');
  $req && return $req;

  if ($dn =~ /^({*,}*,\s*)obj1d=in,(.*)$/) {
    $dn=$1.'obj1d=queue,'.$2;
    ($req)=Get_Obj_Safe(dn=>$dn, scope=>$Scope{base}, type=>'request');
    $req && return $req;
  }

  if ($dn =~ /^({*,}*,\s*)obj1d=queue,(.*)$/) {
    $dn=$1.'obj1d=archive,'.$2;
    ($req)=Get_Obj_Safe(dn=>$dn, scope=>$Scope{base}, type=>'request');
    $req && return $req;
  }

  die "Cannot locate request at $p{dn}, it was moved out or deleted\n"
    unless $p{safe};

  return ();
}

sub Get_JO_Obj {
  my *p = @_ ;
  my $dn = $p{dn};
  my $jo;

  ($jo)=Get_Obj_Safe(dn=>$dn, scope=>$Scope{base}, type=>'joborder');
  $jo && return $jo;

  if ($dn =~ /^({*,}*,\s*[*,}*,\s*)obj1d=queue,(.*)$/) {
    $dn=$1.'obj1d=archive,'.$2;
    ($jo)=Get_Obj_Safe(dn=>$dn, scope=>$Scope{base}, type=>'joborder');
    $jo && return $jo;
  }

  die "Cannot locate Job Order at $p{dn}, it was moved out or deleted\n"
    unless $p{safe};

  return ();
}

sub Display_JO {
  my $dn = shift;

  # get the job order
  my ($jo) = Get_JO_Obj(dn=>$dn);
  $dn = $jo->param('dn');
  #$jo or die("Could not find job order at dn [ $dn ]");

  # figure out request dn
  my $reqdn;
  if ($dn =~ /^(obj1d=[tph\d\.\-|+])\d+,\s*(\1,.*)$/) {
    $reqdn = $2;
  }
  else {
    $dn =~ /^(obj1d=[tph\d\.\-|+])\d+,\s*(.*)$/; obj1d=queue, '.$jo->param('actiondn');
    $reqdn = $1.', obj1d=queue, '.$jo->param('actiondn');
  }

  # get the corresponding request
  my $reqfound=1;
  my ($req) = Get_Obj_Safe(dn=>$reqdn, scope=>$Scope{base}, type=>'request');
  $req or $reqfound=0;
  $req or $req = new CGI("");

  # figure out sequence number
  $dn =~ /^\s*obj1d\s*=\s*[tph\d\.\-|+](\d+),\s*.*$/;
  my $num = $1;

  # figure out if job order is bg or fg
  my $bg = $req->param("joSnum") =~ /^BG / ? 'BG' : ' ';

  # output results
  print CGI_Page
    ($title, 'Job Order '.$jo->param('id'),
```

APPENDIX A
SOURCE CODE LISTING

```
          center
          (font([-size=>-1], "Information as of ", scalar localtime), p,
          table
          ([-border=>5, -cellpadding=>5], Tr
          ([-valign=>'top'],
             [td(['Name:', $jo->param('cn')]),
              td(['ID:', $jo->param('id')]),
              td(['Status:', Color_Status($jo->param('status'))]),
              td(['BG:', $bg]),
              td(['Params:', pre(join("\n", $jo->param('param')))]),
              td(['Returns:', pre(join("\n", $jo->param('rval')))]),
              td(['DN:', $jo->param('dn')]),
              td(['ReqDN:', $reqdn. ($reqfound?'':br."Request not found")]),
              td(['Log:', pre($jo->param('log'))]),
             ])
          ), p,
          submit('s_showreq', 'View Request'),
          hidden('reqdn', $reqdn)
          )
      );
}
sub Handle_Find {
    # search for the object
    my $type = param('type') =~ /^[Rr]equest$/ ? 'request' : 'joborder';
    my @res = Get_Ob)(dn=>param('dn'),
                      scope=>Get_Scope(param('scope')),
                      id=>param('id'),
                      cn=>param('cn'),
                      type=>$type);

    # Nothing found
    die('No matching entries found') if !@res;

    # clean up default query
    Delete_all();

    # Output results
    my $nmatch = $#res + 1;
    print CGI_Page($title,
                   "Found $nmatch matches. Information as of ",
                   scalar localtime,
                   center
                   (Form_Results_List([attrs=>['cn', 'id', 'status', 'dn'],
                                       hide=>['dn', 'type']],
                                      @res))
                  );

}
sub Handle_Display {
    my $dn = param("dn_$_[0]");
    my $type = param("Type_$_[0]");
    Delete_all();
    if ( $type eq 'request' ) {
        &Display_Request($dn);
    } elsif ( $type =~ /^joborder$/i ) {
        &Display_JO($dn);
    } else {
        die("Unknown type of object [ $type ].");
    }
}

sub Display_Request {
    my $dn=$_[0];

    my ($req) = Get_Req_Ob)(dn=>$dn);
    $dn=$req->param('dn');

    # get the request and all related job orders
    my @res = Get_Ob)(dn=>$dn,scope=>$Scope(tree));

    # figure out which one is the request
    #my ($req) = grep $_->param('dn') eq $dn, @res );
    #$req or die("Could not find request at dn [$dn]");
    ($req->param('type') eq 'request') or
        die("Object at dn [ $dn ] is not a request.");

    # figure out job states
    my @js = ();
    my $i;

    for ($i=0;$req->param("jo$i");$i++) { $js[$i] = $req->param("jo$i") }

    # build job order list
    my @jo = ();
    for($i=0; $i<=$#js; $i++) {
        my ($jo) = grep $_->param('id') =~ /j$i$/, @res;
        if ( !$jo ) {
            $jo = new CGI("");
            $jo->param('cn', 'UNKNOWN');
        }
        $jo[$i] = $jo;
    }

    # figure out which jobs are backgrounded
    my ($bg, $status);
    for( $i=0; $i <= $#js; $i++ ) {
        ($bg, $status) = split(' ', $js[$i]);
        $bg = '' if $bg ne 'BG';
        $jo[$i]->param('bg', $bg);
        $jo[$i]->param('status', $status);
```

91

APPENDIX A
SOURCE CODE LISTING

```
)
# hide these values
my @hidden = ( hidden('id', $req->param('id')),
                hidden('dn', $req->param('dn')) );

# output CGI results
print CGI_Page
  ($title, 'Request '.$req->param('id'),
    center
    (font((-size=>-1),'Information as of '. scalar localtime),
     @hidden, p,
     table
     ((-border=>5, -cellpadding=>5), Tr
      ((-valign=>'top'),
        [td(['Name:', $req->param('cn')]),
         td(['ID:', $req->param('id')]),
         td(['Status:', Color_Status($req->param('status'))]),
         td(['PC:', Fill($req->param('pc'))]),
         td(['Params:', pre(join("\n", $req->param('param')))]),
         td(['DN:', $req->param('dn')]),
         td(['Log:', pre($req->param('log'))])]),
       ]
     ),
     p, h2('Job Orders'),
     Form_Results_List((attrs=>['cn', 'status', 'bg'],
                          hide=>['dn', 'type']), @jo)
   )
  );
}
sub Output_Initial_Page {
  print CGI_Page
    (
     $title, 'Welcome to Get Status. Please enter the search criteria',
     table((-cellpadding=>5, -cellspacing=>5), Tr
           ([ td(['Base DN:', textfield(-name=>'dn', -size=>40)]),
              td(['Object ID:', textfield('id')]),
              td(['Object Name:', textfield('cn')]),
              td(['Object Type:', popup_menu('type',
                                   ['Request', 'Job Order'])]),
              td(['Search Scope:', popup_menu('scope', Scope_Text)])
            ])
          ),
     center(submit('s_find', 'Continue')),
    );
}
XXXXXXXXXX END /share/Kiki/WF/prod/perl/get_status XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/perl/run_action XXXXXXXXXX
#!/usr/bin/perl -w

#===========================================================
#
# $Id: run_action,v 1.22 1999/03/03 06:21:07 rt Exp $
#
# Desc: search for action, display it, enter parameters, and
#       run action
#
#===========================================================

# setup
use strict;
use CPAT;
use CPAT::CGI;
use CGI qw/:standard center/;
use vars qw/ $title /;
$title = 'Run Action';

# Main
CGI_Act
  (
   unknown  => \&Output_Initial_Page,
   find     => \&Handle_Find,
   display  => \&Handle_Display,
   run      => \&Handle_Run,
   status   => \&Handle_Status
  );

exit(0);

#===========
# functions
#===========
sub Handle_Status {
  my $id = param('requestid');
  if ( $id =~ /^\s*$/ ) {
    die("Please enter request id");
  }

  # search for request
  my @res = Get_Obj(dn=>$BaseDN, scope=>$Scope(tree), type=>'request',
                    id=>$id);
  die('Could not find the request') if (!@res);

  # figure out which copy of the request we should work with
  my ($r) = grep $_->param('dn') =~ /obj_id=$id, obj_id=in, .*$BaseDN/, @res;
  ($r) = grep $_->param('dn') =~ /obj_id=$id, obj_id=queue, .*$BaseDN/, @res
    if !$r;
  ($r) = grep $_->param('dn') =~ /obj_id=$id, obj_id=archive, .*$BaseDN/, @res
    if !$r;
```

```
die('Could not find a valid copy of request') if (!$r);

# output info about the request
print CGI_Page
  (
   $title, "Status of request $id", p,
   center
   (table
    ({-border=>5, -cellpadding=>5}, Tr
      ([td(['Action Name:', $r->param('cn')]),
        td(['Status:', Color_Status($r->param('status'))]),
        td(['Log:', pre($r->param('log'), ' ')]),
        td(['DN:', $r->param('dn')]),
       ])
    ), p,
    submit('s_status', 'Check Again'), "\n",
    hidden('requestid'), "\n",
   )
  );
}

sub Handle_Run {
  my $dn = param('actiondn');
  ($dn ne '') || die('Need action dn');
  my $bdn = param('binddn') || 'cn=Directory Manager';
  my $bpw = param('bindpw') || 'letsgoskiing';

  # get the action
  my @res = Get_Obj(dn=>$dn, scope=>$Scope(base), type=>'action');
  ($#res == 0) or die("Found $#res+1 matches for dn ( $dn )");

  # check that we have all params
  my @p = $res[0]->param('param');
  my @err = ();
  my @p_vals = ();
  for (@p)
    {
     if (param($_))
       {
        push @p_vals, "_$_", param($_);
       }
     else
       {
        push @err, "Specify value of parameter $_";
       }
    }
  if ( $#err >= 0 )
    {
     die('Your request has not been submitted to LDAP. ' .
         'Please fix the following problems before resubmitting:' . br .
         ul(\@err));
    }
  # submit to ldap
  my $req = Run_Act(dn=>$dn, bdn=>$bdn, bpw=>$bpw, act=>$res[0], @p_vals);

  # output results
  &Output_Submission_Results($req);
}

sub Output_Submission_Results {
  my $req = shift;
  my $id = $req->param('id');

  Delete;
  param('requestid', $id);
  print CGI_Page
    (
     $title, 'Successful Submission',
     "Your request has been submitted with ID ( $id ].", "\n",
     "You can check the status of the request by pressing the button below.",
     "\n", hidden('requestid'), "\n",
     p, center(submit('s_status', 'Check Status'))
    );
}

sub Handle_Display {
  &Display_Entry(param("dn_$_[0]"));
}

sub Display_Entry {
  my $dn=$_[0];
  my @res = Get_Obj(dn=>$dn, scope=>$Scope(base), type=>'action');
  ($#res == 0) or die("Help, found $#res+1 matches for dn $_, ");
  my $res = $res[0];
  my $formurl = $res->param('formurl');
  my $form = $res->param('form');
  my $param = $res->param('param');

  if ($formurl !~ /\s*/) {
    print redirect($formurl);
    return;
  }

  #default form
  my (@vars, $param);
  foreach $param (@param) {
    push @vars, td([$param, textfield($param)]);
  }
  param('actiondn', $dn);
  print CGI_Page($title,
                 'You are about to run ',
```

APPENDIX A
SOURCE CODE LISTING

```perl
                    font({color=>'blue'}, $res->param('cn')),
                    '. Please enter the following parameters:',
                    hidden('actiondn'), p,
                    table(Tr([@vars]]), p,
                    center(submit('s_run', 'Run Action'))
                    );
)

sub Handle_Find {
  my @res = Get_Obj(dn=>$BaseDN,
                    scope=>$Scope{tree},
                    type=>'action',
                    cn=>''.param('actioncn'));

  # Nothing found
  &Output_Nothing_Found() if !@res;

  # Unique entry found - display it
  if ($#res == 0) {
    return &Display_Entry($res[0]->param('dn'));
  }

  # Multiple entries found - display selection list
  &Output_Search_Results(@res);
}
sub Output_Nothing_Found {
  # should put a user friendly page here
  CGI_Page($title, 'No matches found. Please use the back button of your browser and change search criteria');
}
sub Output_Search_Results {
  my @res = @_;
  print CGI_Page($title, 'Found '.($#res+1).' matches',
                    Form_Results_List(@res)
                    );
)

sub Output_Initial_Page {
  print CGI_Page
    (
      $title,
      "Welcome to $title.",

      "To run an action please enter action name",
      center
      (table(Tr(td(['Action Name:', textfield('actioncn'),
                    submit('s_find', 'Continue') ])))
      ),
      p,
      "If you would like to check status of a request you have placed earlier please enter the request id",
      center
      (table(Tr(td(['Request ID:', textfield('requestid'),
                    submit('s_status', 'Check Status') ])))
      ),
      );
)
XXXXXXXXXX END /share/Kiki/WF/prod/perl/run_action XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/perl/CPAT/CGI.pm XXXXXXXXXX
#=========================================================
#
# $Id: CGI.pm,v 1.28 1999/08/24 23:23:45 rt Exp $
#
# Desc: CPAT::CGI - CGI helper functions
#
#=========================================================
package CPAT::CGI;
require Exporter;
@ISA = qw( Exporter );
@EXPORT = qw(CGI_Act CGI_Page Form_Results_List Scope_Text Get_Scope
             Color_Status @Scope_Text Canonical_Type Fill );
use CGI qw/:standard center/;
$SIG{__DIE__} = \&Output_Error;
$Orig_Dump = &CGI::dump;

use vars qw/ @Scope_Text $Orig_Dump /;

@Scope_Text=('Subtree','One Level','Base Only');
{
  my $i;
  my @vals= (2,        1,        0);
  for $i (0..$#Scope_Text) {
    $Scope_Text{$Scope_Text[$i]}=$vals[$i];
  }
}

sub CGI_Act {
  my %Subs = @_;
  my ($cgiact, @cgiparam);

  # figure out cgi action
  ($_)=grep(/^s_/, param,'s_unknown');
  @cgiparam = split /_/;
  shift @cgiparam;
  $cgiact = shift @cgiparam;

  # sanity check
  defined $Subs{$cgiact} ||
    die("Asked to perform unknown cgi action [ $cgiact ]");

  # perform cgi action
  my $subname = $Subs{$cgiact};
  &$subname(@cgiparam);
```

APPENDIX A
SOURCE CODE LISTING

```
sub CGI_Page {
  my $title = shift;
  my $opening = shift;
  my $ret =
    ( header, start_html(-title=>"CPAT: $title", -BGCOLOR=>'white'),

        # outer table
        center
        (table
          ({bgcolor=>"white", cellpadding=>30}, Tr
            (td
              (

                # logo
                center(img({src=>"http://kotya/WF/narrow.gif"})),

                # inner table
                center
                (table
                  ( {-bgcolor=>"white", -cellpadding=>30}, Tr
                    ( td
                      (

                        # Opening message
                        b($opening), p, "\n",

                        # the form
                        "\n", start_form(-method=>'POST', -action=>url()),
                        join("\n",@_), "\n",
                        end_form(), "\n",



                      )))),
                  # end inner table

                  # copyright
                  p,
                  font
                  ({color=>"#6633cc"},
                    center(
                      "Copyright 1999 Anna Petrovskaya,",
                      "11655 Wildflower Ct., Cupertino, CA 95014",
                      br,
                      "All rights reserved"),
                  ),


                )))),
          #end outer table

          end_html
        );
}

sub Output_Error ( ) # does not return!!!
  my $error = shift;
  print CGI_Page
    ( 'CPAT: Error Occured', 'Error Occured',
      p, em(b($error)),
      p, "Original parameters were:", br, $Orig_Dump,
      p, "Current parameters are:", br, &CGI::dump
    );
  # exit(0) needed for web servers
  exit(0);
}

sub Form_Results_List (
  return if $#_ < 0;
  my $p = {};
  $p = shift if ( ref($_[0]) eq 'HASH' );
  my $attrs = ['cn', 'id', 'dm'];
  $attrs = $p->{attrs} if defined $p->{attrs};
  my $hide = ['dm'];
  $hide = $p->{hide} if defined $p->{hide};
  my $px = '';
  $px = $p->{px} if defined $p->{px};
  my $cgiact = 'display';
  $cgiact = $p->{cgiact} if defined $p->{cgiact};
  my (@list, $e, $i, @row, @hidden);
  $i=0;
  foreach $e (@_) {
    @hidden = ();
    for (@$hide) {
      push @hidden, hidden("$(px}$(_)_$i",$e->param($_));
    }
    @row = (center(submit("s_$(cgiact)_$i",$i), @hidden));
    for (@$attrs) {
      if ( $_ eq 'status' ) {
        push @row, Color_Status($e->param($_));
      } elsif ($_ eq 'log') {
        push @row, join($e->param($_), ' ');
      } else {
        push @row, Fill($e->param($_));
      }
    }
    push @list, td(\@row);
```

95

APPENDIX A
SOURCE CODE LISTING

```perl
      $i++;
    }
    @row = ('#');
    for (@$attrs) {
      $_ =~ tr/a-z/A-Z/;
      push @row, $_;
    }
    return table({-border=>5},
                  Tr([th(\@row), @list]));
  }

  sub Scope_Text {
    return \@Scope_Text;
  }

  sub Get_Scope {
    return $Scope_Text[$_[0]];
  }

  sub Color_Status {
    $s = shift;
    my $color = 'red';
    $color = 'green' if ( $s eq 'RUNNABLE' || $s eq 'RUNNING' );
    $color = 'orange' if ( $s eq 'HOLD' );
    $color = 'blue' if ( $s eq 'COMPLETE' );
    $s = 'UNKNOWN' if ( $s =~ /^\s*$/ );
    return font({-color=>$color}, b($s));
  }

  sub Canonical_Type {
    $_ = shift;
    tr/A-Z/a-z/;
    s/\s*//g;
    $_;
  }

  sub Fill {
    my $ret = join("\n", @_, ' ', '');

  };
  XXXXXXXXXX END /share/Kiki/WF/prod/perl/CPAT/CGI.pm XXXXXXXXXX
  XXXXXXXXXX BEGIN /share/Kiki/WF/prod/perl/CPAT/Edit.pm XXXXXXXXXX
  #=============================================================
  #   $Id: Edit.pm,v 1.23 1999/02/03 11:02:30 rt Exp $
  #
  # Desc: CPAT::Edit - Top level edit module
  #
  #=============================================================
  package CPAT::Edit;

  # import stuff
  use strict;
  use CPAT;
  use CPAT::CGI;
  use CGI qw/:standard center/;
  use CPAT::Edit::Main;
  use CPAT::Edit::Action;
  use CPAT::Edit::Agent;
  use CPAT::Edit::Folder;
  use CPAT::Edit::Engine;
  use CPAT::Edit::Job;

  # export stuff
  use vars qw( @ISA @EXPORT );
  require Exporter;
  @ISA = qw( Exporter );
  @EXPORT = qw( @Edit_Functions );

  # setup
  use vars qw( @Edit_Functions );
  @Edit_Functions =
    (
    # for all objects
    unknown    =>    \&Handle_Unknown,
    display    =>    \&Handle_Display,
    new        =>    \&Handle_New,
    commit     =>    \&Handle_Commit,
    exit       =>    \&Handle_Edit,
    revert     =>    \&Handle_Display,
    update     =>    \&Handle_Update,

    # object specific handlers
    @Action_Functions,
    @Agent_Functions,
    @Folder_Functions,
    @Engine_Functions,
    @Job_Functions,
    );

  };
  XXXXXXXXXX END /share/Kiki/WF/prod/perl/CPAT/Edit.pm XXXXXXXXXX
  XXXXXXXXXX BEGIN /share/Kiki/WF/prod/perl/CPAT/Edit/Action.pm XXXXXXXXXX
  #=============================================================
  #   $Id: Action.pm,v 1.15 1999/02/25 02:56:39 rt Exp $
  #
  # Desc: CPAT::Edit::Action - Action functions
  #
```

96

APPENDIX A
SOURCE CODE LISTING

```
#==================================================================
package CPAT::Edit::Action;

# import stuff
use strict;
use CPAT;
use CPAT::CGI;
use CPAT::Edit::Main;
use CGI qw/:standard center/;

# export stuff
use vars qw( @ISA @EXPORT );
require Exporter;
@ISA = qw( Exporter );
@EXPORT = qw( @Action_Functions );

#=========
# setup
#=========
use vars qw( $Obj_Type @Action_Functions @Job_Attrs @Job_Attrs_ %H );
# type of our object
$Obj_Type = 'action';

# add our object to object list
push @Obj_Types, 'Action';

# special cgi action handlers
@Action_Functions =
   (
     actdelline    =>    \&Handle_Del_Line,
     actmvline     =>    \&Begin_Move,
     actmvline2    =>    \&End_Move,
     actinsline    =>    \&Handle_Insert,
     actinsline2   =>    \&Finish_Insert,
     actjobsrch    =>    \&Search_For_Job,
   );

# set up cgi action handlers
$Edit_Map{$Obj_Type} = \&Edit_Action;
$Commit_Map{$Obj_Type} = \&Commit_Action;
$Update_Map{$Obj_Type} = \&Update_Action;
$New_Map{$Obj_Type} = \&New_Action;

# all these need to be defined for each job (script line)
@Job_Attrs = qw/cn dn bg param rval map rmap/;
@Job_Attrs_ = map $_.'_', @Job_Attrs;

# Table Headers
%H = ( num=>'#', job=>'Job', map=>'Param Map', rmap=>'Return Map', bg=>'BG',
       insert=>'Insert', delete=>'Delete', move=>'Move',
       edit=>'Edit' );

#==============
# Functions
#==============
sub Commit_Action {
  my $obj=new CGI('');
  $obj->param('dn', ''.param('dn'));
  $obj->param('cn', ''.param('cn'));
  $obj->param('formurl', ''.param('formurl'));

  # Params
  my @param=split(/\s*,\s*/,param('param'));
  if (@param) {$obj->param('param', @param);}
  else  {$obj->param('param', '');}

  # script
  $obj->param('script',join ('',(map {
    (param("dn_$_"),'{',param("map_$_"),'}',
     '{',param("rmap_$_"),'}',param("bg_$_")?'*':'','',':')
  } 0..param('len')-1)));

  # create new action
  if ( param('new') ) {
    $obj->param('type', ''.Canonical_Type(param('type')));
  }

  # write to LDAP & redisplay
  Put_Obj($obj->$obj, new=>''.param('new'));
  Display_DN( param('dn') );
}

sub New_Action {
  param('len', 0);
  &Update_Action();
}

sub Handle_Insert {
  my $num = shift;
  param('i_to', $num);
  my $jobcn = param('i_cn');
  if ($jobcn) {
    &Job_Search_Page();
  }
  else {
    &Search_For_Job();
  }
}
```

APPENDIX A
SOURCE CODE LISTING

```perl
sub Finish_Insert {
  my $srnum = shift;
  my $insrtnum = param('i_to');
  my ($objdn = param("i_res_dn_$srnum");
  my ($job) = Get_Obj(dn=>''.$jobdn, scope=>$$Scope(base));

  grep param("i_$(_)_", $job->param($_)),qv(cn dn param rval);
  &Insert_Line($insrtnum);
  &Update_Action();
}

sub Job_Search_Page {
  print CGI_Page
    ($Title, 'Search for a job to insert into script',
      center(table
              ([Tr(
                  [td(['Job Name:', textfield(-name=>'i_cn', -size=>40)]),
                   td(['Base DN:', textfield(-name=>'i_dn', -size=>40)]),
                   td(['Scope:', popup_menu('i_scope', \@Scope_Text)])
                  ])
                ),
                submit('s_actjobsrch', 'Search'),
              ),
      Hide_All_Vars('i_cn')
    );
}

sub Search_For_Job {
  my $scope = $$Scope(tree);
  $scope = Get_Scope(param('i_scope')) if param('i_scope');
  my @res = Get_Obj(dn=>''.param('i_dn'),
                    cn=>''.param('i_cn'),
                    type=>'job',
                    scope=>$scope);
  @res or die("No matching entries found");
  print CGI_Page
    ($Title, 'Found '.scalar @res.' matches',
      Form_Results_List((cqisct=>'actinsline2', px=>'i_res_'), @res),
      Hide_All_Vars('i_scope', 'i_cn', 'i_dn'),
    );
}

# TODO:
# should still check and fix up syntax of fields
sub Update_Action {
  my $ret = '';
  my $len = param('len');
  my @ap = split(/\s*,\s*/, param('param'));
  my ($i, %ap, @jp, %jp, %jpd, $jcn, $name, $val);

  # reread all jobs
  for( $i=0; $i<$len; $i++) {
    my ($job) = Get_Obj(dn=>''.param("dn_$i"), scope=>$$Scope(base), type=>'job');
    if ($job) {grep param("$(_)_$i",$job->param($_)),qv(cn dn param rval);}
    else {
      param("cn_$i",'UNKNOWN');
      param("param_$i", '');
      $ret .= "Could not retrieve job $i [ ".param("dn_$i")." ]".br;
    }
  }

  # build action param hash - for quick access
  grep $ap($_)=1, @ap;

  # check all params of all jobs are
  # mapped to something in action params
  for( $i=0; $i<$len; $i++)
    {
      @jp = split(/,\s*/, param("param_$i"));
      %jp = ();
      %jpd = ();
      $jcn = param("cn_$i");
      grep $jpd($_)++,param("param_$i");
      for (@jp) {
        ($name, $sname) = /(.*?)\s*=\s*(.*)/;
        if($jpd($name)) or $ret.="No parameter [ $name ] in job [ $i:$jcn ]".br;
        for ($val =~ /\s<(.*?)>/g) {
          ($ap($_)) or $ret.="No action parameter [ $_ ] referenced from parameter [ $name ] in job [ $i:$jcn ]".br;
        }
        $jp($name) = $val;
      }
      @jp = param("param_$i");
      for (@jp) {
        next if (exists $jp($_) || exists $ap($_));
        next if (/^\s*$/);
        $ret .= "Parameter [ $_ ] of job [ $i:$jcn ] does not have a value".br;
      }
    }

  # check all returns of all jobs are
  # mapped to something in action params
  for( $i=0; $i<$len; $i++)
    {
      @jp = split(/,\s*/, param("rmap_$i"));
      %jp = ();
      %jpd = ();
      $jcn = param("cn_$i");
      grep $jpd($_)++,param("rval_$i");
      for (@jp) {
        ($name, $val) = /(.*?)\s*=\s*(.*)/;
```

APPENDIX A
SOURCE CODE LISTING

```perl
        ($ap{$name}) or $ret.="No action parameter [ $name ] for job return [ $i:$jcn ]".br;
        for ($val =~ /\+<(.*?)>/g) {
        ($pd[$_]) or $ret.="No job return [ $_ ] referenced for action parameter [ $name ] in job [ $i:$jcn ]".br;
        }
        $jp{$name} = $val;
    }
  }

  # return string of errors (html format)
  if ( $ret ne '' )
    {
      $ret = comment('Errors') . p . font
        ({-color=>'red'},
         b('Errors:'), blockquote($ret)
        ) . p;
    }

  &Display_Action($ret);
}

# errors (if any) may be passed as the first arg
sub Display_Action {
  my $errors = shift;
  print CGI_Page
    ($Title, 'Editing '.font({color=>'blue'}, param('cn')),
     $errors, "\n",
     &Action_Info_Table(), "\n",
     &Editable_Script(), "\n",
     &Job_Lookup_Field(), "\n",
     &Action_Buttons(), "\n",
     &Action_Hidden_Vars(), "\n",
    );
}

sub Edit_Action {
  my $act = shift;
  Delete_all();
  param('dn', $act->param('dn'));
  param('cn', $act->param('cn'));
  param('type', $act->param('type'));
  param('param', join(", ", $act->param('param')));
  param('formurl', $act->param('formurl'));
  &Parse_Script($act->param('script'));
  &Update_Action();
}

sub Parse_Script {
  my $script = shift;
  my @lines = split(/\s*;\s*/, $script);
  param('len', $#lines+1);
  my $i = 0;
  my ($job, $map, $rmap);
  for( @lines ) {
    /^\s*(.*?)\s*((\(\s*(.*?)\s*\))|(\(\s*(.*?)\s*\))?)?\s*(\4)?\s*$/;
    param("dn_$i", $1);
    $7 && param("bg_$i", 'on');
    param("map_$i", ''.$4);
    param("rmap_$i", ''.$6);
    $i++;
  }
}

sub Handle_Del_Line {
  my $num = shift;
  &Delete_Line($num);
  &Update_Action();
}

sub Begin_Move {
  my $num = shift;
  param('m_from', $num);
  &Print_Move_Page($num);
}

sub End_Move {
  my $num = shift;
  my $from = param('m_from');
  my $to = $num;
  my $len = param('len');

  $to-- if ( $to > $from );

  # save off the line (used by insert)
  grep param("i_$_", param("$_$from")), @Job_Attrs;

  &Delete_Line($from);
  &Insert_Line($to);
  &Update_Action();
}

# delete a line from script
# line number passed as first arg
sub Delete_Line {
  my $n = shift;
  my $len = param('len');
  my ($i, $j);

  # delete the line
  grep Delete("$_$n"), @Job_Attrs;

  # move up lines following the line
```

APPENDIX A
SOURCE CODE LISTING

```perl
for($i=$n-1; $i<$len; $i++)
  {
    $j = $i - 1;
    for (@Job_Attrs_) {
        param("$_$j", param("$_$i"));
        Delete("$_$i");
    }
  }

# update script length
$len--;
param('len', $len);


# insert a line into script at location passed as first arg
# contents of the line is passed as i_* parameters of CGI
# Insert_Line will delete i_* parameters at the end
sub Insert_Line {
  my $n = shift;
  my $len = param('len');
  my ($i, $j);

  # move down lines following line $n
  for($i=$len-1; $i>=$n; $i--)
    {
      $j = $i + 1;
      for (@Job_Attrs_) {
          param("$_$j", param("$_$i"));
          Delete("$_$i");
      }
    }

  # insert the line and clean up i_* params
  for (@Job_Attrs_) {
    param("$_$n", param("i_$_"));
    Delete("i_$_");
  }

  # update script length
  $len++;
  param('len', $len);
}

sub Action_Info_Table {
  my @ret =
    (comment('Action Info table'),
     center
     (table
      (Tr
       ([td(['Action Name:', textfield(-name=>'cn', -size=>40)]),
         td(['Action DN:', Fill(param('dn'))]),
         td(['Params:', textfield(-name=>'param', -size=>40)]),
         td(['Form URL:', textfield(-name=>'formurl', -size=>40)]),
         td(['Script:', ''])
        ])
      )),
    );
}

sub Editable_Script {
  my $len = param('len');
  my $i;
  my @script;
  my @hdrs = ($H{num}, $H{job}, $H{bg}, $H{map}, $H{rmap},
              $H{insert}, $H{move}, $H{delete});
  push @script, th(\@hdrs);
  for( $i=0; $i<$len; $i++)
    {
      push @script, td
        ([$i,
          e(-href=>url()."?type=job&s_edit=x&dn=".
            &CGI::escape(param("dn_$i")),
            -target=>"_TOP"),
            Fill(param("cn_$i")),
          checkbox(-name=>"bg_$i", -label=>''),

          textfield(-name=>"map_$i",-size=>16).br.
          font((-size=>-1),'Params: '.(join(', ', param("perem_$i")))),

          textfield(-name=>"rmap_$i",-size=>8).br.
          font((-size=>-1),'Returns: '.(join(', ', param("rvel_$i")))),

          submit("s_actinsline_$i", 'Insert'),
          submit("s_actmvline_$i", 'Move'),
          submit("s_actdelline_$i", 'Delete'),
         ]);
    }
  push @script, td(\-colspan=>$#hdrs+1),
                     center(submit("s_actinsline_$len", $H{insert})));
  my @ret =
    (comment('Editable Script'),
     center(table((-cellspacing=>5, -cellpadding=>10, -border=>5),
                  Tr(\@script))));
}

sub Job_Lookup_Field {
  my @ret =
    (comment('Job Lookup field'),
     center('Insert Job Named: ',textfield(-name=>'i_cn', -size=>20)));
}
```

APPENDIX A
SOURCE CODE LISTING

```perl
sub Action_Buttons {
  my @ret =
    (comment('Action Buttons'),
    p,table
     ({-width=>'100%'},Tr
       (td(submit('s_update', 'Update View')),
        td({-align=>'center'}, submit('s_commit', 'Commit Action')),
        td({-align=>'right'}, submit('s_revert', 'Revert Action'))
       )
      )
     );
}

sub Action_Hidden_Vars {
  my $len = param('len');
  my $i;
  my @hidden_vars;

  # global hidden vars
  push @hidden_vars, hidden('len'), hidden('dn'), hidden('type'), hidden('new');

  # script hidden vars
  for( $i=0; $i<$len; $i++)
    {
      for (@Job_Attrs_) {
          next if (/^bg_$/); # bg is a checkbox in editable script
          next if (/^map_$/); # map is editable
          next if (/^rmap_$/); # rmap is editable
          push @hidden_vars, hidden("$_$i");
        }
     }
  my @ret = join("\n",(comment('Hidden Vars'),@hidden_vars));
}

# move from is passed as first arg
sub Print_Move_Page {
  my $n = shift;
  print CGI_Page
    ($Title, 'Moving job '.font({color=>'blue'}, param("cn_$n")),
     &Movable_Script($n), "\n",
     &Hide_All_Vars(), "\n",
     );
}

# move from is passed as first arg
sub Movable_Script {
  my $n = shift;
  my $len = param('len');
  my $i;
  my @script;
  my $place = 'Place Here';
  my $cancel = 'Cancel Move';

  push @script, Tr([$H{num}, $H{job}, $H{map}, $H{rmap}]);
  for( $i=0; $i<$len; $i++)
    {
      if ( $i == $n ) {
          push @script,
          td({-colspan=>4, -align=>'center'},
             submit("s_update", $cancel)),
          td({-bgcolor=>'yellow'},
             [$i, Fill(param("cn_$i")), Fill(param("map_$i")), Fill(param("rmap_$i"))]);
        }
      elsif ( $i == $n+1 ) {
          push @script,
          td({-colspan=>4, -align=>'center'},
             submit("s_update", $cancel)),
          td([$i, Fill(param("cn_$i")), Fill(param("map_$i")), Fill(param("rmap_$i"))]);
        }
      else {
          push @script,
          td({-colspan=>4, -align=>'center'},
             submit("s_actmvline2_$i", $place)),
          td([$i, Fill(param("cn_$i")),Fill(param("map_$i")), Fill(param("rmap_$i"))]);
        }
     }
  if ( $len > $n+1 ) {
      push @script, td({-colspan=>4, -align=>'center'},
                       submit("s_actmvline2_$len", $place));
   }
  else {
      push @script, td({-colspan=>4, -align=>'center'},
                       submit("s_update", $cancel));
   }
  my @ret =
    (comment('Movable Script'),
     table({-cellspacing=>5, -cellpadding=>10, -border=>5}, Tr([@script])),
     );
}
#;
XXXXXXXXXX END /share/Kiki/WF/prod/perl/CPAT/Edit/Action.pm XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/perl/CPAT/Edit/Agent.pm XXXXXXXXXX
#------------------------------------------------------------
#
#  $Id: Agent.pm,v 1.4 1999/02/25 02:10:38 rt Exp $
#
#  Desc: CPAT::Edit::Agent - Agent functions
#
```

APPENDIX A
SOURCE CODE LISTING

```perl
#==================================================================
package CPAT::Edit::Agent;

# import stuff
use strict;
use CPAT;
use CPAT::CGI;
use CPAT::Edit::Main;
use CGI qw(:standard center);

# export stuff
use vars qw( @ISA @EXPORT );
require Exporter;
@ISA = qw( Exporter );
@EXPORT = qw( @Agent_Functions );

#-------
# setup
#-------
use vars qw( $Obj_Type @Agent_Functions );

# type of our object
$Obj_Type = 'agent';

# add our object to object list
push @Obj_Types, 'Agent';

# no special cgi actions
@Agent_Functions = ();

# set up cgi action handlers
$Edit_Map{$Obj_Type} = \&Edit_Agent;
$Commit_Map{$Obj_Type} = \&Commit_Agent;
$Update_Map{$Obj_Type} = \&Display_Agent;
$New_Map{$Obj_Type} = \&Display_Agent;


#============
# Functions
#============
sub Edit_Agent {
  my $obj = shift;
  Delete_all();
  grep { param($_, '',$obj->param($_}); } qw/ dn cn type /;
  &Display_Agent();
}
sub Display_Agent {
  my $comments = shift;
  print CGI_Page
     ($Title, 'Editing '.font((color=>'blue'), param('cn')),
      $comments, p,
      center(table
             (Tr
              ([td(['Agent Name:', textfield(-name=>'cn', -size=>40)])])),
             table
             (Tr
              (td([submit('s_commit', 'Commit Agent'),
                    submit('s_revert', 'Revert Agent')])))),
      hidden('new'), hidden('dn'), hidden('type')
     );
}
sub Commit_Agent {
  my $obj=new CGI('');

  # dn, cn
  $obj->param('dn', ''.param('dn'));
  $obj->param('cn', ''.param('cn'));

  # create new agent
  if ( param('new') ) {
    $obj->param('type', Canonical_Type(param('type')));
  }

  # write to LDAP & redisplay
  Put_Obj(obj=>$obj, new=>''.param('new'));
  Display_DN( param('dn') );
}

1;
XXXXXXXXXX END /share/Kiki/WF/prod/perl/CPAT/Edit/Agent.pm XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/perl/CPAT/Edit/Engine.pm XXXXXXXXXX
#==================================================================
#
# $Id: Engine.pm,v 1.4 1999/02/25 02:11:34 rt Exp $
#
# Desc: CPAT::Edit::Engine - Engine functions
#
#==================================================================
package CPAT::Edit::Engine;

# import stuff
use strict;
use CPAT;
use CPAT::CGI;
use CPAT::Edit::Main;
use CGI qw(:standard center);

# export stuff
use vars qw( @ISA @EXPORT );
require Exporter;
```

APPENDIX A
SOURCE CODE LISTING

```
@ISA = qw( Exporter );
@EXPORT = qw( @Engine_Functions );

#-------
# setup
#-------
use vars qw( $Obj_Type @Engine_Functions );

# type of our object
$Obj_Type = 'engine';

# add our object to object list
push @Obj_Types, 'Engine';

# no special cgi actions
@Engine_Functions = ();

# set up cgi action handlers
$Edit_Map{$Obj_Type} = \&Edit_Engine;
$Commit_Map{$Obj_Type} = \&Commit_Engine;
$Update_Map{$Obj_Type} = \&Display_Engine;
$New_Map{$Obj_Type} = \&Display_Engine;


#============
# Functions
#============
sub Edit_Engine {
  my $obj = shift;
  Delete_all();
  grep { param($_, '', $obj->param($_)); } qw/ dn cn type /;
  my @actdn = $obj->param('actiondn');
  my $num = $#actdn + 1;
  param('num', $num);
  my $i;
  for ($i=0; $i<$num; $i++) {
    param("actiondn_$i", $actdn[$i]);
  }
  &Display_Engine();
}

sub Display_Engine {
  my $comments = shift;
  print CGI_Page
    ($Title, 'Editing '.font({color=>'blue', param('cn')),
     $comments, p,
     center(table
       (Tr
         ([td(['Engine Name:', textfield(-name=>'cn', -size=>40)])]),
        table
        (Tr
         (td([submit('s_commit', 'Commit Engine'),
              submit('s_revert', 'Revert Engine')])))))),
     Hide_All_Vars(qw/ cn /),
     );
}

sub Commit_Engine {
  my $obj=new CGI('');

  # dn, cn
  $obj->param('dn', '', param('dn'));
  $obj->param('cn', '', param('cn'));

  # actiondn
  my $num = param('num');
  if ( $num == 0 ) { $obj->param('actiondn', ''); }
  else {
    my @actdn = ();
    my $i;
    for ($i=0; $i<$num; $i++) {
      push @actdn, param("actiondn_$i"),
    }
    $obj->param('actiondn', @actdn);
  }

  # create new engine
  if ( param('new') ) {
    $obj->param('type', Canonical_Type(param('type')));
  }

  # write to LDAP & redisplay
  Put_Obj(obj=>$obj, new=>''.param('new'));
  Display_DN( param('dn') );
}

1;
XXXXXXXXXX END /share/Kiki/WF/prod/perl/CPAT/Edit/Engine.pm XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/perl/CPAT/Edit/Folder.pm XXXXXXXXXX
#==============================================================
#
#  $Id: Folder.pm,v 1.9 1999/02/25 02:12:43 rt Exp $
#
# Desc: CPAT::Edit::Folder - Folder functions
#
#==============================================================
package CPAT::Edit::Folder;

# import stuff
use strict;
use CPAT;
use CPAT::CGI;
```

APPENDIX A
SOURCE CODE LISTING

```
use CPAT::Edit::Main;
use CGI qw/:standard center/;

# export stuff
use vars qw( @ISA @EXPORT );
require Exporter;
@ISA = qw( Exporter );
@EXPORT = qw( @Folder_Functions );

#-------
# setup
#-------
use vars qw( $Obj_Type @Folder_Functions );

# type of our object
$Obj_Type = 'folder';

# add our object to object list
push @Obj_Types, 'Folder';

# no special cgi actions
@Folder_Functions = ();

# set up cgi action handlers
$Edit_Map{$Obj_Type} = \&Edit_Folder;
$Commit_Map{$Obj_Type} = \&Commit_Folder;
$Update_Map{$Obj_Type} = \&Display_Folder;
$New_Map{$Obj_Type} = \&Display_Folder;


#============
# Functions
#============
sub Edit_Folder {
  my $obj = shift;
  Delete_all();
  grep { param($_, ''.$obj->param($_)); } qw/ dn cn type /;
  &Display_Folder();
}
sub Display_Folder {
  my $comments = shift;
  print CGI_Page
    ($Title, 'Editing '.font({color=>'blue'}, param('cn')),
     $comments, p,
     center(table
              (Tr
                 ([td(['Folder Name:', textfield(-name=>'cn', -size=>40)])]),
                table
                  (Tr
                     (td([submit('s_commit', 'Commit Folder'),
                          submit('s_revert', 'Revert Folder')]))))),
        hidden('new'), hidden('dn'), hidden('type')
     );
}
sub Commit_Folder {
  my $obj=new CGI('');

  # dn, cn
  $obj->param('dn', ''.param('dn'));
  $obj->param('cn', ''.param('cn'));

  # create new folder
  if ( param('new') ) {
    $obj->param('type', Canonical_Type(param('type')));
  }

  # write to LDAP & redisplay
  Put_Obj($obj=>$obj, new=>''.param('new'));
  Display_DN( param('dn') );
}

1;
XXXXXXXXXX END /share/Kiki/WF/prod/perl/CPAT/Edit/Folder.pm XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/perl/CPAT/Edit/Job.pm XXXXXXXXXX
#=====================================================================
#
# $Id: Job.pm,v 1.11 1999/02/25 02:13:54 rt Exp $
#
# Desc: CPAT::Edit::Job - Job functions
#
#=====================================================================
package CPAT::Edit::Job;

# import stuff
use strict;
use CPAT;
use CPAT::CGI;
use CPAT::Edit::Main;
use CGI qw/:standard center/;

# export stuff
use vars qw( @ISA @EXPORT );
require Exporter;
@ISA = qw( Exporter );
@EXPORT = qw( @Job_Functions );

#-------
# setup
#-------
use vars qw( @Job_Functions $Obj_Type );
```

104

APPENDIX A
SOURCE CODE LISTING

```perl
# type of our object
$Obj_Type = 'job';

# add our object to object list
push @Obj_Types, 'job';

# no special cgi actions
@Job_Functions = ();

# set up cgi action handlers
$Edit_Map{$Obj_Type} = \&Edit_Job;
$Commit_Map{$Obj_Type} = \&Commit_Job;
$Update_Map{$Obj_Type} = \&Display_Job;
$New_Map{$Obj_Type} = \&Display_Job;


#============
# Functions
#============
sub Commit_Job {
  my $obj=new CGI('');

  # dn, cn, agent and command
  $obj->param('dn', ''.param('dn'));
  $obj->param('cn', ''.param('cn'));
  $obj->param('agentdn', ''.param('agentdn'));
  $obj->param('command', ''.param('command'));

  # param
  my @param=split(/\s*,\s*/,param('param'));
  $obj->param('param', (@param)?@param:'');

  # rval
  my @rval=split(/\s*,\s*/,param('rval'));
  $obj->param('rval', (@rval)?@rval:'');

  # create new job
  if ( param('new') ) {
    $obj->param('type', Canonical_Type(param('type')));
  };

  # write to LDAP & redisplay
  Put_Obj(obj=>$obj, new=>''.param('new'));
  Display_DN( param('dn') );
}

sub Edit_Job {
  my $job = shift;
  Delete_all();
  param('dn', $job->param('dn'));
  param('cn', $job->param('cn'));
  param('type', $job->param('type'));
  param('param', join(", ", $job->param('param')));
  param('rval', join(", ", $job->param('rval')));
  param('agentdn', $job->param('agentDN'));
  param('command', $job->param('command'));
  Display_Job();
}

sub Display_Job {
  my $comments = shift;
    print CGI_Page
      ( $Title, 'Editing '.font({color=>'blue'}, param('cn')),
        p, font({color=>'red'}, $comments),
        p, table
          (Tr
          ([td(['Job Name:', textfield(-name=>'cn', -size=>60)]),
            td(['Job DN:', Fill(param('dn'))]),
            td(['Agent DN:', textfield(-name=>'agentdn', -size=>60)]),
            td(['Command:', textfield(-name=>'command', -size=>60)]),
            td(['Params:', textfield(-name=>'param', -size=>60)]),
            td(['Returns:', textfield(-name=>'rval', -size=>60)])
          ])),
        p, table
          ({-width=>'100%'}, Tr
            (td({-align=>'left'}, submit('s_commit', 'Commit Job')),
             td({-align=>'right'}, submit('s_revert', 'Revert Job')))),
        hidden('type'), hidden('dn'), hidden('new')
      );
}

1;
XXXXXXXXXX END /share/Kiki/WF/prod/perl/CPAT/Edit/Job.pm XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/perl/CPAT/Edit/Main.pm XXXXXXXXXX
#====================================================================
#
# $Id: Main.pm,v 1.12 1999/02/25 02:01:34 rt Exp $
#
# Desc: CPAT::Edit::Main - Main functions
#
#====================================================================
package CPAT::Edit::Main;

# import stuff
use strict;
use CPAT;
use CPAT::CGI;
use CGI qw/:standard center/;
```

APPENDIX A
SOURCE CODE LISTING

```
# export stuff
use vars qw( @ISA @EXPORT );
require Exporter;
@ISA = qw( Exporter );
@EXPORT = qw( %Edit_Map %Commit_Map %Update_Map %New_Map @Obj_Types
              Handle_Edit Handle_Commit Handle_Display Handle_Update
              Handle_New Handle_Unknown Print_Entrance_Page
              Hide_All_Vars Output_Search_Results Display_DN
              Display_Obj $Title
            );

# setup
use vars qw( %Edit_Map %Commit_Map %Update_Map %New_Map @Obj_Types $Title );
%Edit_Map = ();
%Commit_Map = ();
%Update_Map = ();
%New_Map = ();
@Obj_Types = ();
$Title = 'Edit Object';

#============
# functions
#============
sub Handle_Display {
  my $dn;
  if ( @_ ) { $dn = param('dn_'.$_[0]); }
  else { $dn = param('dn'); }
  Display_DN($dn);
}

sub Display_DN {
  my $dn = shift;

  # get the object
  my ($obj) = Get_Obj(dn=>''.$dn, scope=>$Scope(base));
  $obj or die("Could not retrieve object at dn [ $dn ]");

  Display_Obj( $obj );
}

sub Display_Obj {
  my $obj = shift;

  # check that we have handler defined
  my $type = $obj->param('type');
  defined $Edit_Map($type) ||
    die("No edit handler defined for type [ $type ]");

  # run the handler
  my $subname = $Edit_Map($type);
  &$subname($obj);
}

sub Handle_New {
  # figure out dn
  my $dn = param('dn');
  if ( $dn !~ /$BaseDNS/  ||
       $dn !~ /^objid=\w+,/ ) { die("Malformed dn [ $dn ]"); }

  # Check to see if the object already exists
  my ($o) = Get_Obj(dn=>''.$dn, scope=>$Scope(base), safe=>1);
  if ( ref $o ) { die("Object at dn [ $dn ] exists"); }
  if ( $o !~ /no such object/i ) { die( 'Error Message: '.$o ); }

  # Ok, it's safe to make new object
  param('new', '1');

  # check that we have handler defined
  my $type = Canonical_Type(param('type'));
  defined $New_Map($type) ||
    die("No new handler defined for type [ $type ]");

  # run the handler
  my $subname = $New_Map($type);
  &$subname();
}

sub Handle_Commit {
  my $type = Canonical_Type(param('type'));
  defined $Commit_Map($type) ||
    die("No commit handler defined for type [ $type ]");

  # run the handler
  my $subname = $Commit_Map($type);
  &$subname();
}

sub Handle_Edit {
  my $type = Canonical_Type(param('type'));
  $type or die("Need to know type of object to edit");
  my $scope = Get_Scope(param('scope'));
  my $dn = param('dn');
  $dn !~ /^\s*$/ or die("Need a non-empty base dn");
  my @res = Get_Obj(dn=>''.$dn,
                    scope=>''.$scope,
                    cn=>''.param('cn'),
                    type=>''.$type);

  # Nothing found
```

APPENDIX A
SOURCE CODE LISTING

```perl
    die('No matching entries found') if !@res;

    # Multiple entries found - display selection list
    if (@#res > 0) (return Output_Search_Results(@res))

    # Unique entry found - display it
    Display_Obj( $res[0] );
}

sub Handle_Update (
    my $type = Canonical_Type(param('type'));
    defined $Update_Map{$type} ||
        die("No update handler defined for type [ $type ]");

    # run the handler
    my $subname = $Update_Map{$type};
    &$subname();
)

sub Handle_Unknown (
    Delete_all();
    param('dn', $BaseDN);
    &Print_Entrance_Page();
)

sub Print_Entrance_Page (
    print CGI_Page
        ($Title, 'Welcome to Edit Object',
        'Please specify search criteria for the object',
        'you would like to edit', br,
        b(em('Note:')), 'you do not have to fill out all fields.', "\n",
        center(table
            ((cellpadding=>5), Tr
                ([td(['Object Name:', textfield(-name=>'cn', -size=>40)]),
                    td(['Object Type:', br, font([color=>'red', size=>'-1'],
                                                    '(required)')]),
                            popup_menu('type', \@Obj_Types)]),
                    td([Fill('Base DN:', br, font([color=>'red', size=>'-1'],
                                                    '(required)')]),
                            textfield(-name=>'dn', -size=>40)]),
                    td(['Scope:', popup_menu('scope', \@Scope_Text)])
                    ])
                ), p,
            table
            ((width=>'75%'), Tr
                (td(submit('s_new', 'New')),
                td((align=>'right'), submit('s_edit', 'Edit'))
                ))
            )
        );
)

sub Output_Search_Results (
    my @res = @_;
    print CGI_Page($Title,
                    'Found '.(@#res+1).' matches, ',
                    'Information as of '. scalar localtime,
                    Form_Results_List(@res),
                    );
)

sub Hide_All_Vars (
    my @hidden_vars;
    my %skip = ();
    for (@_) (
        $skip{$_} = 1;
    )

    for (param) (
        next if (/^s_/); # do not pass submit vars
        next if (exists $skip{$_}); # skip vars passed as args
        push @hidden_vars, hidden("$_");
    )
    my @ret = join("\n",comment('Hidden Vars'),@hidden_vars);
)

1;
XXXXXXXXXX END /share/Kiki/WF/prod/perl/CPAT/Edit/Main.pm XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/engine/cfg XXXXXXXXXX
# system config file
cfg=/share/Kiki/WF/prod/syscfg

# engine log file
log=/share/Kiki/WF/prod/engine/log

# engine's DN and password
#bdn=objid=enginel, objid=TOP, o=NONE
#bpw=helloworld

# subtree to service
servicedn=objid=TOP, o=NONE

# set once for now
#once=true

XXXXXXXXXX END /share/Kiki/WF/prod/engine/cfg XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/agent/cfg XXXXXXXXXX
# system config
cfg=/share/Kiki/WF/prod/syscfg

# agent log file
```

APPENDIX A
SOURCE CODE LISTING

```
log=/share/Kiki/WF/prod/agent/log

# agent's record
servicedn=objid=agntl, objid=TOP, o=NONE

# agent's DN and password
#bdn=objid=agentl, objid=TOP, o=NONE
#bpw=helloworld

#once=y


XXXXXXXXXX END /share/Kiki/WF/prod/agent/cfg XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/agent/Makefile XXXXXXXXXX
# $Id: Makefile,v 1.5 1999/02/13 19:57:D9 root Exp $

OBJS = Pipe_IO.o Job_Run.o child.o AgentD.o
BINARIES = agentd

# Major targets
all: cleanup $(BINARIES)
release: all
          cp agentd $(BINDIR)
agentd: $(OBJS)

# Objects
Pipe_IO.o: Pipe_IO.h
AgentD.o: AgentD.h Job_Run.h
Job_Run.o: Job_Run.h Pipe_IO.h
child.o: AgentD.h

# Cleanup targets
cleanup:
          rm -f *~ core

clean: cleanup
          rm -f *.o $(BINARIES)


XXXXXXXXXX END /share/Kiki/WF/prod/agent/Makefile XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/engine/Makefile XXXXXXXXXX
# $Id: Makefile,v 1.3 1999/02/13 19:57:4D root Exp $

OBJS = EngineD.o
BINARIES = engined

# Major targets
all: cleanup $(BINARIES)
release: all
          cp engined $(BINDIR)
engined: $(OBJS)

# Objects
EngineD.o: EngineD.h

# Cleanup targets
cleanup:
          rm -f *~ core

clean: cleanup
          rm -f *.o $(BINARIES)


XXXXXXXXXX END /share/Kiki/WF/prod/engine/Makefile XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/ui/Makefile XXXXXXXXXX
# $Id: Makefile,v 1.11 1999/02/13 19:54:59 root Exp $

BINARIES = get_obj run_action update_obj move_obj

# Major targets
all: cleanup $(BINARIES)
release: all
          cp get_obj $(BINDIR)
          cp run_action $(BINDIR)
          cp update_obj $(BINDIR)
          cp move_obj $(BINDIR)

# Cleanup targets
cleanup:
          rm -f *~ core

clean: cleanup
          rm -f *.o $(BINARIES)


XXXXXXXXXX END /share/Kiki/WF/prod/ui/Makefile XXXXXXXXXX
XXXXXXXXXX BEGIN /share/Kiki/WF/prod/util/Makefile XXXXXXXXXX
# $Id: Makefile,v 1.12 1999/02/15 07:25:07 rt Exp $

OBJS = Exception.o LDAP_related.o Util.o Params.o LDAP_Entry.o
OBJS += LDAP_Wrap.o Obj.o Job_State.o Request.o Job_Order.o Action.o
OBJS += Script.o Req_Builder.o

# Major targets
all: cleanup libwf.a
release: all
          cp libwf.a $(LIBDIR)

# Util library
libwf.a: $(OBJS)
```

APPENDIX A
SOURCE CODE LISTING

```
        rm -f $@
        $(AR) cqv $@ $^

# Objects
Params.o: Params.h
Util.o: Util.h
LDAP_Wrap.o: LDAP_Wrap.h
LDAP_Entry.o: LDAP_Entry.h
LDAP_related.o: LDAP_related.h
Exception.o: Exception.h
Action.o: Action.h
Job_State.o: Job_State.h
Request.o: Request.h
Obj.o: Obj.h
Job_Order.o: Job_Order.h
Req_Builder.o: Req_Builder.h
Script.o: Script.h

# Cleanup targets
cleanup:
        rm -f *~ core

clean: cleanup
        rm -f *.o *.so *.a

XXXXXXXXXX END /share/Kiki/WF/prod/util/Makefile XXXXXXXXXX
```

WHAT IS CLAIMED IS:

1.   A system to coordinate the execution of a plurality of separate target computer systems to effectuate a process, the system comprising:

(a) a core system for receiving a request by a user to effectuate the process, the request including user data upon which it is desired to effectuate the process and an indication of an action corresponding to the process; and

(b) the core system further including a centralized execution controller that controls and coordinates execution of the target computer systems based on the user data and execution rules corresponding to the indicated action, thereby accomplishing effectuation of the process.

2.   The system of claim 1, wherein:

(a) the target computer systems do not communicate with each other about the execution thereof.

3.   The system of claim 1, wherein:

(a) the core system includes an authenticator that authenticates the user and determines, based on the authentication, whether the user is authorized for the process; and

(b) the execution controller operates in part based on the authorization determination.

4.   The system of claim 1, wherein:

(a) the system further comprises:

(i) a monitor that monitors execution of the target computer systems and logs monitoring results thereof; and

(ii) a log processor that processes the monitoring results based on a query from the user.

110

5.   The system of claim 1, wherein:

(a) the core system includes a memory that holds, in a non-volatile manner, a state of execution of the target computer systems; and

(b) the core system operates based on the state of execution stored in the memory.

6.   The system of claim 1, and further comprising:

(a) a configurator that receives indications of process definition information from a developer and provides the execution rules based on the process definition information.

7.   The system of claim 6, wherein:

(a) the configurator displays to the developer a plurality of possible indications of the process definition information; and

(b) at least some of the indications received by the configurator from the developer are a subset of the plurality of possible indications displayed by the configurator.

8.   The system of claim 6, wherein:

(a) the configurator includes means for creating a user interface and relating the execution rules to the user interface; and

(b) the core system operates at least in part based on interaction by the user with the user interface.

9.   The system of claim 8, wherein:

(a) the user interface is a first user interface; and

(b) the configurator includes means for creating a second user interface based on at least a portion of the first user interface.

111

10.   The system of claim 6, wherein:

(a) the process is a first process; and

(b) the configurator includes means for providing at least some of the execution rules, for effectuating the first process, to effectuate the second process.

11.   The system of claim 1, wherein the core system includes:

(a) a data store that stores the execution rules; and

(b) an engine that operates on the stored execution rules to effectuate the process.

12.   The system of claim 11, wherein:

(a) the data store further stores an indication of a state of execution of the target computers; and

(b) the engine further operates on the stored execution state.

13.   The system of claim 1, wherein:

(a) the centralized execution controller comprises agent means, executing on the target computer systems, for causing the target computer systems to execute based on the execution rules to effectuate the process.

14.   The system of claim 13, wherein:

(a) the agent means includes monitoring means for monitoring the execution of the target computer systems and for generating monitoring results thereof; and

(b) the engine means operates at least in part based on the monitoring results.

15.   The system of claim 13, wherein:

(a) at least one of the target computer systems executes an operating system; and

(b) the agent means executing on that target computer system controls execution of the target computer system by making at least one system call to the operating system.

16.   The system of claim 1, wherein:

(a) the core system includes a plurality of components; and

(b) the system further comprises a data store via which the components of the core system communicate.

17.   The system of claim 16, wherein:

(a) the core system includes a component that is a user interface for interaction with at least one user; and

(b) the user interface stores data into the data store based on actions of the user for communication with other components of the core system.

18.   The system of claim 17, wherein the user interface further retrieves data from the data store for communicating the information to the user.

19.   The system of claim 1, wherein:

(a) the core system operates based on data objects; and

(b) the system further comprises means for creating new data objects based on old data objects, whereby the old data objects are reusable.

20.   The system of claim 19, wherein:

(a) the data objects include jobs and job orders;

(b) the core system operates on the job orders; and

(c) the system includes means for creating the job orders based on the jobs.

21.   The system of claim 20, wherein:

(a) the data objects include actions;

(b) the actions are comprised of the jobs.

5   22.   The system of claim 21, wherein:

(a) the actions are further comprised of user interface component templates;

(b) the data objects include user interface component templates; and

(c) the core system interacts with the user based at least in part on the user interface component templates.

10

23.   The system of claim 19, wherein:

(a) the data objects include actions and requests;

(b) the core system operates on the requests; and

15   (c) the system includes means for creating the requests based on the actions.

24.   The system of claim 19, wherein:

(a) the data objects include user interface component templates; and

20   (b) the core system interacts with the user based at least in part on the user interface component templates.

25.   A system to coordinate the execution of a plurality of separate target computer systems to effectuate a process, the system comprising:

25   (a) user interface means for receiving a request by a user to effectuate the process, the request including user data upon which it is desired to effectuate the process and an indication of an action corresponding to the process;

(b) engine means for generating a series of job orders based on the user data and execution rules corresponding to the indicated action; and

30   (c) agent means for causing execution of the target computer systems based on the job orders, thereby accomplishing effectuation of the process.

114

26. The system of claim 25, wherein:

(a) the execution rules include target computer system dependent commands; and

(b) the engine modifies the target computer dependent commands

5   based on the user-supplied data and provides the modified target computer dependent commands to the agents so that the target computer systems accomplish the effectuation of the business process in a specific manner corresponding to the user-supplied data.

10   27. The system of claim 26, wherein:

(a) the modified target computer dependent commands are part of the job orders.

28. The system of claim 26, wherein:

15   (a) the job orders include pointers to the modified target computer dependent commands.

29. The system of claim 25, and further including:

(a) storage means for storing the actions and requests.

20

30. The system of claim 25, and further including:

(a) storage means for storing the generated execution results and from which the generated execution results are accessible to the engine means.

25   31. The system of claim 29, wherein the storage means is a directory server.

32. The system of claim 25, and further comprising:

(a) storage means for storing the request provided by the user.

30

33.   The system of claim 32, wherein:

(a) the storage means is coupled to the engine means for storing the series of job orders; and

(b) the storage means is further coupled to the agent means for providing the job orders to the agent means.

34.   The system of claim 25, wherein:

(a) each action has corresponding to it:

(i)   an input means into which requests from a user are stored before being provided to the engine means;

(ii)   a job order storage means into which the job orders are stored; and

(iii) an output means into which status indicators are stored after attempted effectuation of the business process.

35.   The system of claim 34, wherein:

(a) the engine means includes:

(i)   means for receiving the request from the input means corresponding to the action;

(ii)   means for processing the request received from the input means to generate the job orders; and

(iii) means for providing the generated job orders to the job order storage means; and

(b) the agent means includes means for receiving the generated job orders from the job order storage means.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(54) Title: SYSTEM FOR DEVELOPMENT AND MAINTENANCE OF SOFTWARE SOLUTIONS FOR EXECUTION ON DISTRIBUTED COMPUTER SYSTEMS
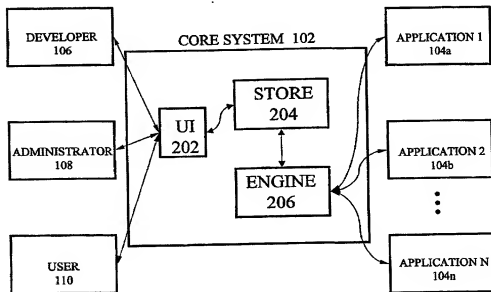
(57) Abstract: A system is provided to effectuate steps of a process such as a business process. A core system receives a request by a user to effectuate the process, along with user data upon which it is desired to effectuate the process. A coordinating system causes and coordinates execution of a plurality of target computer systems based on the indication of the action and user data, to accomplish effectuation of the process.

Fig. 1

CORE SYSTEM 102

APPLICATION 1
104a

APPLICATION 2
104b

· · ·

APPLICATION N
104n

STORE
204

ENGINE
206

UI
202

DEVELOPER
106

ADMINISTRATOR
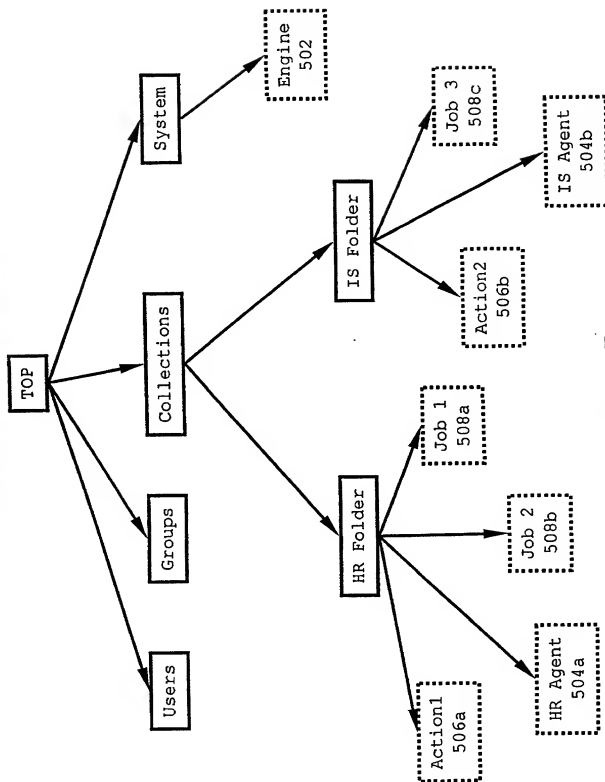108

USER
110

Fig. 2

Fig. 3

10/088949

Fig. 4

Fig. 5

Fig. 6

10/088949
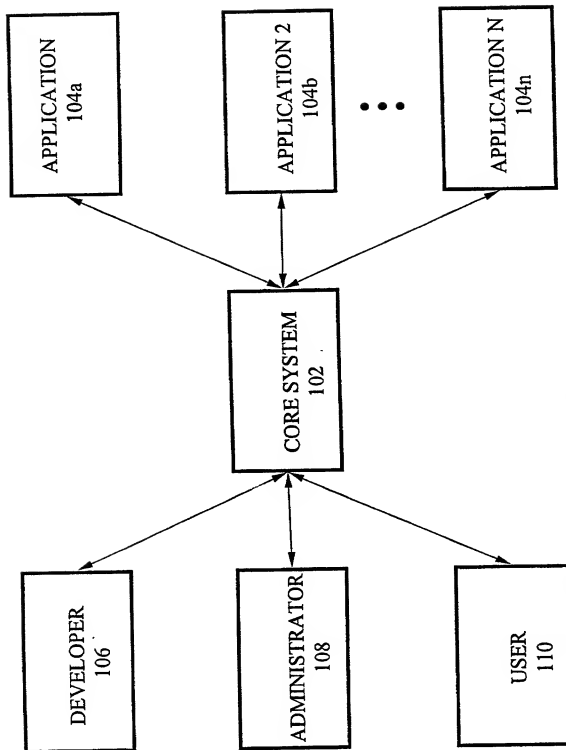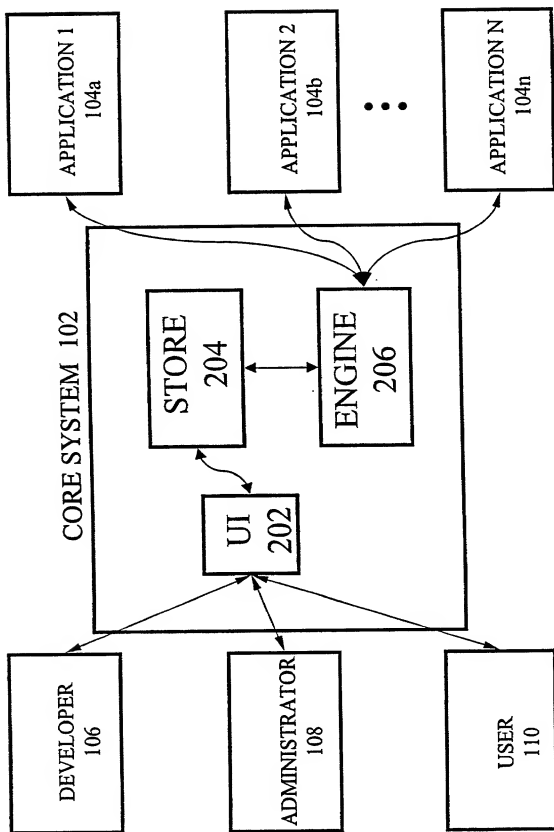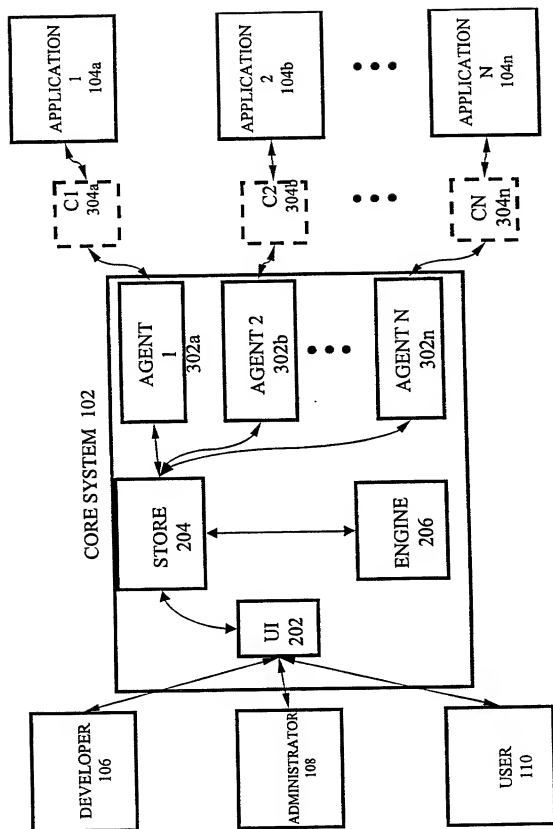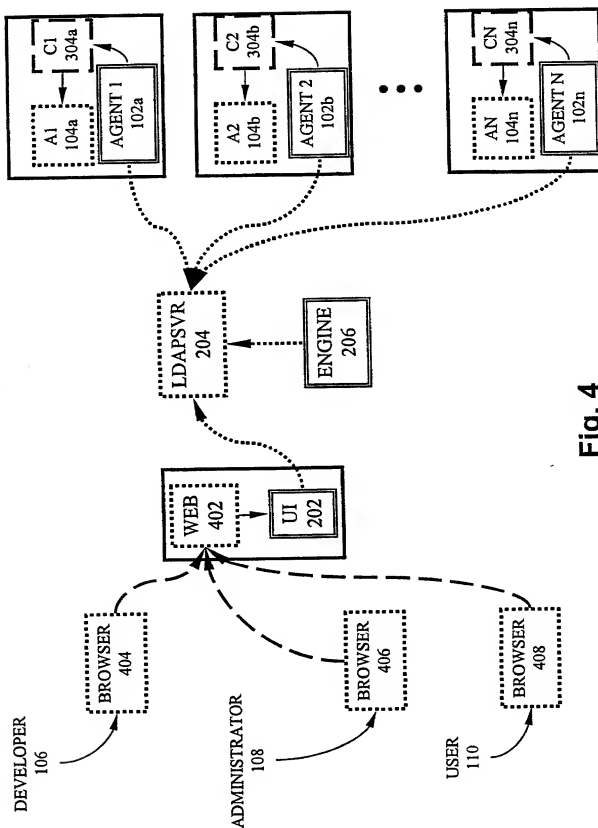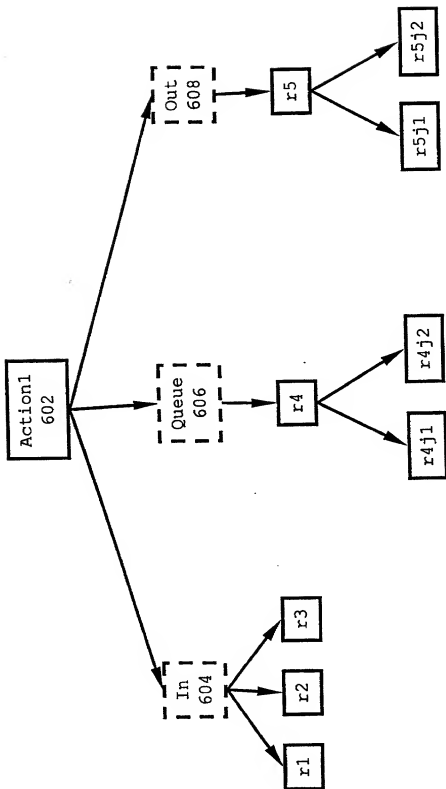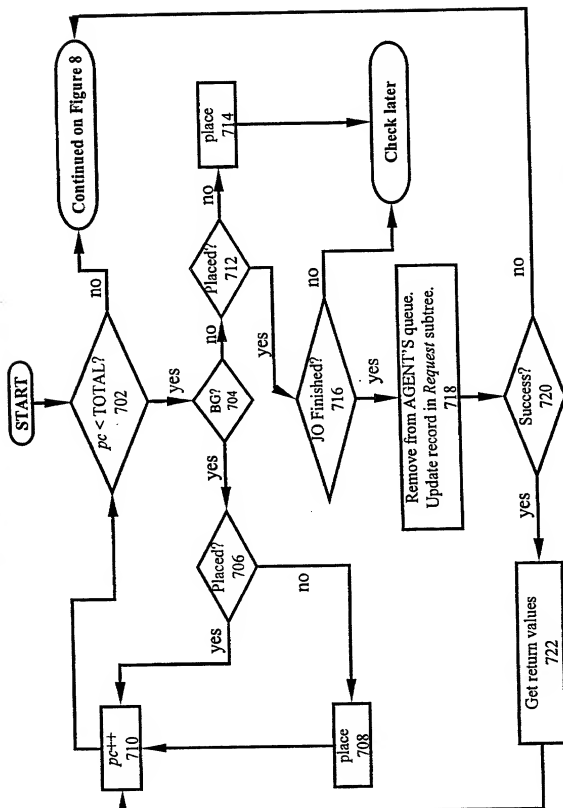
Fig. 7

BG = background *Job Order*
TOTAL = total number of *Job Orders* in *Request*
JO = *Job Order*

10/088949

Continued from Figure 7

→ Cycle through all BGs and remove finished JOs from AGENTS' queues.
802

Are all BGs done?
804

no → Check later.

yes → Determine *status* of *Request*.
806

Move *Request* subtree from Queue to Out.
808

Done with this *Request*.
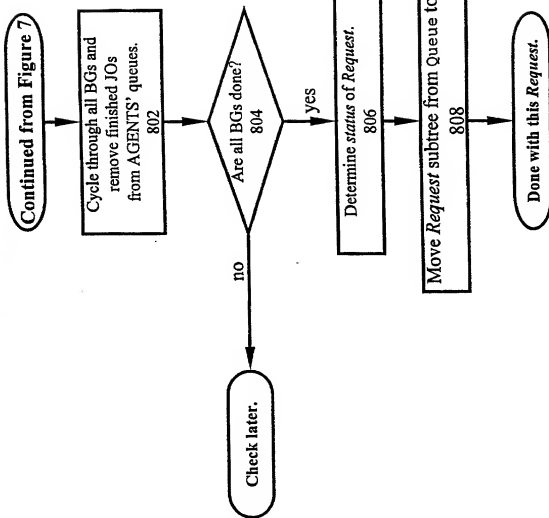
Fig. 8

BG = background *Job Order*
JO = *Job Order*

PATENT
Docket No. 514592000100

DECLARATION FOR PATENT APPLICATION	ATTORNEY'S DOCKET NUMBER
(Includes Reference to PCT International Applications)	514592000100

As a below named inventor I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name,

I believe I am the original, first and sole inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled:

SYSTEM TO COORDINATE THE EXECUTION OF A PLURALITY OF SEPARATE COMPUTER SYSTEMS TO EFFECTUATE A PROCESS (AMENDED IN SEARCH REPORT)

the specification of which (check only one item below):

☐ is attached hereto.

☐ was filed as United States application

Serial No. To Be Assigned
on To Be Assigned,
and was amended on * (if applicable).

☒ was filed as PCT international application

Number PCT/US00/26631
on September 28, 2000

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37 Code of Federal Regulations § 1.56(a) and (b).

I hereby claim foreign priority benefits under Title 35 United States Code § 119 of any foreign application(s) for patent or inventor's certificate or of any PCT international application(s) designating at least one country other than the United States of America listed below and have also identified below any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date before that of the application(s) of which priority is claimed:

PRIOR FOREIGN/PCT APPLICATION(S) AND ANY PRIORITY CLAIMS UNDER 35 U.S.C. § 119:

COUNTRY (if PCT indicate "PCT")	APPLICATION NUMBER	DATE OF FILING (day, month, year)	PRIORITY CLAIMED UNDER 35 U.S.C. § 119
U.S.	60/156,809	September 29, 1999	☒ YES ☐ NO
			☐ YES ☐ NO
			☐ YES ☐ NO
			☐ YES ☐ NO
			☐ YES ☐ NO

1

Declaration for Patent Application (Continued)	ATTORNEY'S DOCKET NUMBER
(Includes Reference to PCT International Applications)	514592000100

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) or PCT international application(s) designating the United States of America that is/are listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in that/those prior application(s) in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, § 1.56(a) which occurred between the filing date of the prior application(s) and the national or PCT international filing date of this application:

PRIOR U.S. APPLICATIONS OR PCT INTERNATIONAL APPLICATIONS DESIGNATING THE U.S. FOR BENEFIT UNDER 35 U.S.C. § 120:

U.S. APPLICATIONS		STATUS (Check one)		
U S APPLICATION NUMBER	U S FILING DATE	PATENTED	PENDING	ABANDONED

PCT APPLICATIONS DESIGNATING THE U.S.			STATUS (Check one)		
PCT APPLICATION NUMBER	PCT FILING DATE	U S SERIAL NUMBERS ASSIGNED (if any)	PATENTED	PENDING	ABANDONED

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith. (List name and registration number)

Send correspondence to:	Direct telephone calls to:
Alan S. Hodes Morrison & Foerster LLP 755 Page Mill Road Palo Alto, California 94304-1018	Alan S Hodes at (650) 813-5622

201	FULL NAME OF INVENTOR	FAMILY NAME PETROVSKAYA	FIRST GIVEN NAME Anna	SECOND GIVEN NAME
	RESIDENCE & CITIZENSHIP	CITY Cupertino	STATE OR FOREIGN COUNTRY California	COUNTRY OF CITIZENSHIP United States of America
	POST OFFICE ADDRESS	11655 Wildflower Court	CITY Cupertino	STATE & ZIP CODE/COUNTRY California 95014
202	FULL NAME OF INVENTOR	FAMILY NAME	FIRST GIVEN NAME	SECOND GIVEN NAME
	RESIDENCE & CITIZENSHIP	CITY	STATE OR FOREIGN COUNTRY	COUNTRY OF CITIZENSHIP
	POST OFFICE ADDRESS	POST OFFICE ADDRESS	CITY	STATE & ZIP CODE/COUNTRY
203	FULL NAME OF INVENTOR	FAMILY NAME	FIRST GIVEN NAME	SECOND GIVEN NAME
	RESIDENCE & CITIZENSHIP	CITY	STATE OR FOREIGN COUNTRY	COUNTRY OF CITIZENSHIP
	POST OFFICE ADDRESS	POST OFFICE ADDRESS	CITY	STATE & ZIP CODE/COUNTRY

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

SIGNATURE OF INVENTOR 201	SIGNATURE OF INVENTOR 202	SIGNATURE OF INVENTOR 203
DATE 3-21-02	DATE	DATE

2